UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería de
Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Desarrollo de un sistema de análisis de registros de flujos de red

Ricardo Domingo Curt
Tutor: Jorge Enrique López de Vergara Méndez

JULIO 2018

Desarrollo de un sistema de análisis de registros de flujos de red

AUTOR: Ricardo Domingo Curt
TUTOR: Jorge Enrique López de Vergara Méndez

High Performance Computing and Networking Research Group
Dpto. de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio de 2018

Resumen

La monitorización de red se suele basar en el uso de sondas que capturan el tráfico paquete a paquete y generan registros. Por ejemplo, el software M^{3}OMON, desarrollado en el grupo HPCN-UAM, a partir del tráfico capturado proporciona registros de los flujos unidireccionales asociados a los paquetes analizados y guarda la información más relevante: fecha, duración del flujo, direcciones y puertos, y los primeros bytes del nivel de aplicación de dicho flujo. En este TFG se plantea la explotación de dichos registros mediante su análisis, tratando de obtener información adicional del nivel de aplicación, como métodos utilizados, direcciones visitadas, códigos de respuesta, etc. Para ello será necesario analizar la información almacenada en registros ya disponibles, de forma que se pueda enriquecer con datos del nivel de aplicación, distinguiendo para cada protocolo.

Se tratan dos protocolos en concreto: **HTTP** (*Hypertext Transfer Protocol*) y **DNS** (*Domain Name System*). Ambos protocolos tienen en común el hecho de que, si se analizan a bajo nivel, se pueden extraer estadísticas sobre el funcionamiento de la red bajo estudio. Para exprimir al máximo la información que se puede obtener de estos protocolos y debido a las diferencias existentes entre los protocolos mencionados, en este proyecto se desarrollan dos herramientas de análisis distintas: **HTTP *Flow Analyzer***, que será la encargada de analizar el protocolo HTTP y, **DNS *Flow Analyzer***, que se encargará de analizar el protocolo DNS.

Para mostrar el funcionamiento de estos programas, se ha llevado a cabo una validación comparando los datos que las herramientas desarrolladas han sido capaces de analizar frente a herramientas ya existentes como HTTP *Dissector* y *Parser* DNS.

Palabras clave

Tráfico, Análisis, HTTP, DNS, Registros, Flujos de Red, Protocolo.

Abstract

Network monitoring is usually based on the use of probes that capture packet-by-packet traffic and generate logs. For example, the M3OMON software, developed in the HPCN-UAM group, from the captured traffic provides records of the unidirectional flows associated with the analyzed packets and stores the most relevant information: date, duration of flow, addresses and ports, and the first bytes of the application level of that flow. This TFG considers the exploitation of these registers by means of their analysis, trying to obtain additional information on the level of application, such as methods used, addresses visited, response codes, etc. To do this, it will be necessary to analyze the information stored in already available records, so that it can be enriched with data from the application level, distinguishing for each protocol.

Two protocols are dealt with in particular: HTTP (Hypertext Transfer Protocol) and DNS (Domain Name System). Both protocols have in common the fact that, if analyzed at a low level, statistics on the operation of the network under study can be extracted. In order to squeeze the maximum information that can be obtained from these protocols and due to the differences between the protocols mentioned, two different analysis tools are developed in this project: HTTP *Flow Analyzer*, which will be in charge of analyzing the HTTP protocol and DNS *Flow Analyzer*, which will be in charge of analyzing the DNS protocol.

To show how these programs work, a validation has been carried out comparing the data that the tools developed have been able to analyze against existing tools such as HTTP Dissector and Parser DNS.

Keywords

Traffic, Data Analysis, HTTP, DNS, Records, Network Flows, Protocol.

Agradecimientos

Existen, en la vida, situaciones que nos ponen a prueba, situaciones en las que a veces creemos que incluso llegamos al límite de nuestro esfuerzo y es entonces cuando hay que tener claro que "No Conseguirás Grandes Resultados Sin Esfuerzo" y como si esta frase fuera mía te dedico a ti, ***Papá****, todo el trabajo que este proyecto implica. Te dedico a ti,* ***Mamá****, mi esfuerzo diario, mis ganas por superarme y aprovecho para pedirte perdón por pasarme mañanas, tardes, días y noches en la universidad sin poder disfrutar todo lo que me hubiera gustado a tu lado. Y a ti,* ***Cris****, te dedico la ilusión, ganas y te transmito toda la fuerza del mundo para que alcances tus objetivos este y el resto de los años que te quedan de vida universitaria. Todo este proyecto es para vosotros* ***Family****. Es un orgullo, un placer, una suerte y un verdadero honor ser vuestro hijo y os agradezco hoy y todos los días la paciencia, el esfuerzo, la sonrisa y todos los momentos que me regaláis.*

A vosotros, porque sin vosotros hubiera sido imposible alcanzar la meta. Vosotros que hacéis de cada madrugón un DesayUam, vosotros que convertís los momentos más difíciles en momentos memorables y que habéis conseguido crear una familia en 4 años. David Abreu, como no podía ser de otra manera, podemos por fin decir que lo hemos conseguido y sabes que me llena de emoción haber terminado contigo. Antuán Clamarán Supreme, compañero, amigo y hermano de derrapes, derrapes que no solo hemos cometido en los karts. No me han enseñado a contar tantos números como para alcanzar el número de horas que hemos pasado juntos en la 24 horas. Sergio Vivas, déjame que te tome de referencia, eres el más grande de los 4 y cada año te superas, no hay quien te pare. Espero que sigas transmitiéndonos tu actitud y que sigas guiándonos durante innumerables años más. Gracias de corazón Official Ludo's.

Gracias por los viajes, fiestas, fútbol y risas que me has regalado jgalan10. Este año hemos crecido como personas y juntos como amigos, no sé qué hubiera hecho, sobre todo, este último tramo de curso sin esas conversaciones que me daban aliento y ganas para terminar este curso y me motivaban a hacer cosas nuevas y no estancarme. JORSS90 y pablitofs90 es siempre un placer pasar buenos ratos con vosotros. Gracias por haberme enseñado si uno quiere, tiene tiempo para todo en esta vida.

Abuelo Ricardo, es momento de decirlo: ¡Aire, Aire!
Estoy seguro de que las velas que la abuela Julia ha puesto cada vez que he tenido un examen llevaban con ellas parte de ti, parte de tu espíritu y tu fuerza. Sin vosotros no hubiera llegado tan lejos.

Jorge E. López de Vergara, durante este año has sido mi tutor, mi guía. Ha sido una experiencia verdaderamente enriquecedora trabajar a tu lado, he aprendido muchas cosas durante todas las tutorías que hemos tenido. Ya no sé ni cuantas veces te lo he dicho, pero, de verdad millones de gracias por tu paciencia.

Sin terminar aquí ¡A por la milla extra!

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Glosario

- **HTTP:** *Hypertext Transfer Protocol*. Protocolo de Transferencia de Hipertexto.
- **DNS:** *Domain Name System*. Sistema de Nombres de Dominio.
- **TCP:** *Transport Control Protocol*. Protocolo de Control de Transporte.
- **TS:** *Timestamp.* Marca temporal.
- **IP:** *Internet Protocol*. Protocolo de Internet.
- **RA:** *Recursion Available*. Recursión Disponible.
- **RD:** *Recursion Desired*. Recursión Deseada.
- **AA:** *Authoritative Answer.* Respuesta Autoritativa.
- **TC:** *Truncation.* Truncado.
- **CDF:** *Cumulative Distribution Function*. Función de Distribución Acumulada.
- **ICMP:** *Internet Control Message Protocol*. Protocolo de Control de Mensajes de Internet.
- **SSL:** *Secure Socket Layers.* Capa de Sockets Seguros.
- **IPFIX:** *IP Flow Information Export*. Exportación de Información de Flujo IP.
- **IANA:** *Internet Assigned Numbers Authority.* Autoridad de Internet para la Asignación de Números.
- **CSV:** *Comma Separated Values*. Extensión de archivos con valores separados por comas.

1 Introducción

1.1 Motivación

El análisis de tráfico de red adquiere cada día más importancia desde que los avances tecnológicos que hoy acontecen y modifican la forma en la que se vive generan un volumen ingente de datos, del cual es verdaderamente útil explotar ciertos campos que son muy relevantes para la sociedad, las distintas entidades empresariales, tanto públicas como privadas, hospitales, etc.

Si bien se habla del tráfico de red, cabe destacar el aumento exponencial de usuarios que Internet ha vivido. Tomando como referencia el año 1995, en el que se llegaron a superar los 15 millones de usuarios activos, no cabía esperar que a principios de este mismo año 2018, el número de usuarios activos fuera de 4.157 millones, lo que supone un 54.4% de la población mundial según reflejan los datos obtenidos de [16]. Esta evolución se puede notar en la FIGURA 1-1.

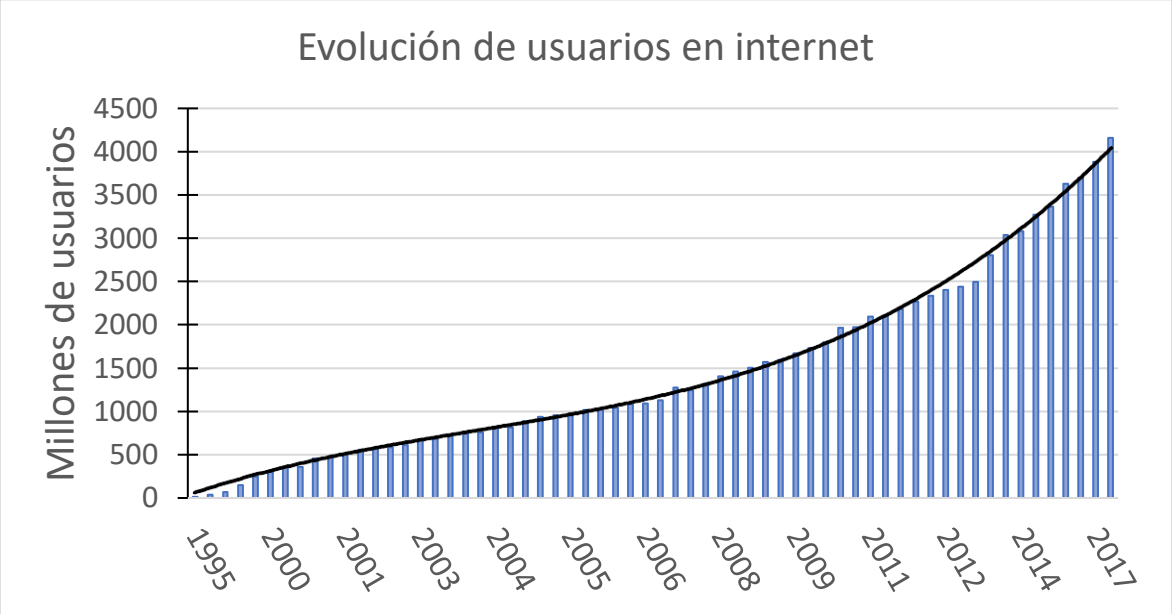

FIGURA 1-1: Evolución de usuarios en internet a lo largo del tiempo según [16].

Una vez se llega a este punto, es necesario analizar los flujos de datos que circulan por la red, para optimizar el funcionamiento que esta tiene, o desde otro punto de vista más protector, para ver si los procesos que se ejecutan en nuestras redes personales, a nivel particular o empresarial, son los que realmente se quieren. Puede que haya intrusiones (no tienen por qué ser humanas) en las redes que ralenticen la forma en que estas funcionan, y la única forma de evitar que esto ocurra es precisamente el análisis de todos estos datos.

1.2 Objetivos y enfoque

El objetivo general es el desarrollo de herramientas que nos permitan analizar registros de flujos de red, en vez de paquetes, dado que de esta forma se es capaz de monitorizar el funcionamiento de la red y de obtener información útil sobre el tráfico que circula por ella. Los objetivos y enfoque de este trabajo están claramente divididos en 2 partes: análisis del protocolo HTTP sobre flujos de red y análisis del protocolo DNS sobre flujos de red.

1.2.1 Objetivos del HTTP *Flow Analyzer* y del DNS *Flow Analyzer*.

- Analizar sobre dos archivos abiertos en paralelo, de registros y de carga útil, distintos registros de flujos de red del protocolo HTTP.
- Presentar nuevas soluciones para mejorar las herramientas de análisis actual.
- Que dicha herramienta disponga de un ciclo de vida actualizable, de tal forma que, en un futuro, se puedan seguir realizando mejoras sobre el diseño.
- Realizar una serie de pruebas/validaciones que acrediten el desarrollo efectivo seguido.

1.3 Metodología y plan de trabajo.

La realización de este trabajo ha comenzado con una primera parte de documentación, en la cual se han tenido en cuenta distintos aspectos y necesidades a cumplir para después llevar a cabo un diseño consecuente. Para ello, se ha realizado un análisis de distintas opciones para satisfacer de forma efectiva los requisitos.

El desarrollo seguido ha permitido ir perfeccionando las herramientas desarrolladas, mientras iban apareciendo nuevos retos, debido a que existen distintas necesidades para analizar protocolos diferentes. En primer lugar, se desarrolló una herramienta que realizaba un análisis detallado del protocolo HTTP (*Hipertext Transfer Protocol*), obteniendo información que puede ser utilizada para monitorizar el rendimiento de la red. El objetivo es explotar la información que el protocolo HTTP puede proporcionar como métodos, códigos de respuesta, direcciones visitadas, etc.

En segundo lugar, se realizó un desarrollo (exhaustivo y más costoso en cuanto a tiempo se refiere) sobre el protocolo DNS (Domain Name System). Con esta herramienta se puede verificar cómo se asocian los nombres de dominio dentro de una red y ver qué tipo y clase de preguntas y respuestas se llevan a cabo en todo momento. Se muestra a continuación, un diagrama de Gantt que detalla la metodología y planificación seguidas hasta el fin del proyecto, como se muestra en la FIGURA 1-2:

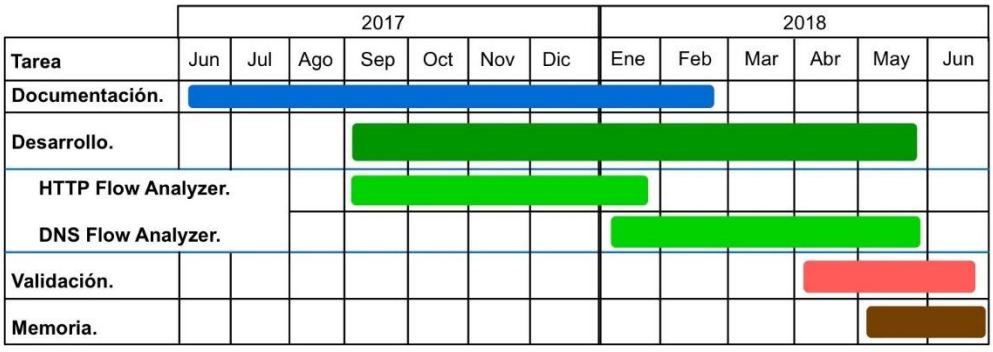

FIGURA 1-2: Diagrama de Gantt

1.4 Organización del documento

Para poder presentar los resultados derivados del trabajo realizado y mostrar cómo se han alcanzado los objetivos finales, este documento consta de los siguientes capítulos:

- **Capítulo 2: Estudio del Estado del arte.** Durante este capítulo se profundiza en las bases que fundamentan este proyecto. Se darán también una serie de razones por las cuales se ha escogido un lenguaje de programación concreto y será analizado el impacto del formato de los registros de trabajo. Se incluirá una descripción sobre los protocolos protagonistas en este TFG (HTTP y DNS). Para concluir se enumerarán una serie de razones que describen la necesidad de analizar registros de flujos de red y se aprovechará para realizar una pequeña comparación de las herramientas existentes actualmente frente a las herramientas desarrolladas en este TFG.

- **Capítulo 3: Diseño.** Se verá en esta sección una primera explicación de por qué C ha sido elegido el lenguaje de programación utilizado para llevar a cabo este proyecto. Este capítulo aborda el estudio de los registros con los que se ha trabajado, para dejar clara su estructura y formato, así como la relación entre los dos archivos que deben ser abiertos en paralelo proporcionados para el análisis de registros de flujos de red. En segundo lugar, se describe cómo ha sido pensada la estructura de la herramienta que analiza flujos con protocolo HTTP, la cual, a partir de ahora será nombrada **HTTP *Flow Analyzer***. De forma análoga se verá el efecto que tiene la estructura de la herramienta que analiza el protocolo DNS sobre el análisis de los flujos. De ahora en adelante, esta herramienta será llamada **DNS *Flow Analyzer***.

- **Capítulo 4: Desarrollo.** Es en este capítulo donde se detallan las etapas que se han seguido para abordar el análisis de flujos de red. Se introduce el concepto fundamental para este proyecto de **flujos de red simétricos**. En primera instancia, se presenta el HTTP *Flow Analyzer* y el desarrollo tanto la disección de los registros como una descripción del método llevado a cabo para resolver el análisis en varias etapas que permiten alcanzar los objetivos marcados.
  Asimismo, se describe la forma en la que se ha desarrollado el DNS *Flow Analyzer*, el cual, debido al formato de los datos descrito en el capítulo anterior, debe de ser pensado para procesar los flujos de distinta forma que el caso del protocolo HTTP.

- **Capítulo 5: Integración, pruebas y resultados.** Este capítulo contiene los resultados específicos para los datos obtenidos a partir de los registros de cada protocolo, orientados a la validación del trabajo llevado a cabo. Es en esta sección cuando se detalla la utilidad de las herramientas desarrolladas.

- **Capítulo 6: Conclusiones y trabajo futuro.** En este capítulo quedan sintetizadas las conclusiones extraídas a partir del trabajo realizado y se presentan posibles líneas de trabajo futuro para extender el estudio realizado. En última instancia se encuentran los anexos. En concreto, el anexo A que detalla la estructura que siguen los registros de los flujos de red.

2 Estado del arte

2.1 Introducción

En este capítulo se presenta una base de estudios previos que motivan y apoyan la realización este proyecto. Para estructurar el estado del arte, en primer lugar, se proyectan las ideas principales que fundamentan el protocolo HTTP. En segundo lugar, se ahonda en el protocolo DNS y los campos que constituyen su cabecera. Esto permite obtener una visión general acerca de los protocolos que durante este TFG han sido objeto de estudio. A continuación, se detalla el formato de los registros de flujos de red con los que se ha estado trabajando para comprender la forma en la que deben ser tratados. Para concluir, se extraerán las ideas principales y serán descritas algunas empresas de carácter multinacional que llevan a cabo monitorización del tráfico de red a través de registros de flujos.

2.2 Protocolo HTTP

El protocolo HTTP (*Hypertext Transfer Protocol*) como se detalla en [1] permite la solicitud y transferencia de documentos web. HTTP es un ejemplo de protocolo de dominio público que define el formato y secuencia de los mensajes que se pasan entre el navegador y el servidor web. Por esto, HTTP es una pieza muy importante de las aplicaciones y servicios web.

HTTP se implementa habitualmente en dos programas: un cliente y un servidor. Ambos se ejecutan simultáneamente en dos sistemas terminales diferentes y se comunican entre sí intercambiando mensajes. La estructura de estos mensajes queda definida por el protocolo de la capa de aplicación, así como la forma en la que cliente y servidor se comunican.

HTTP utiliza TCP (*Transport Control Protocol*), la descripción detallada se presenta en [2], como protocolo de transporte subyacente, de tal forma que, cuando un usuario solicita una página web, el navegador envía al servidor mensajes de solicitud HTTP. Una vez el servidor responde a esa solicitud, lo hace en este caso con códigos de respuesta que son traducidos, de forma que se sabe si se ha producido algún fallo durante la transacción entre cliente y servidor.

Lo interesante de este protocolo para el análisis de registros de flujos de red es que este tipo de peticiones son registradas en texto plano. Esto supone una gran ventaja para el análisis detallado de los registros que contienen la información sobre la conexión. A continuación, la FIGURA 2-1 muestra una pequeña descripción de una conversación simple entre cliente y servidor de HTTP.

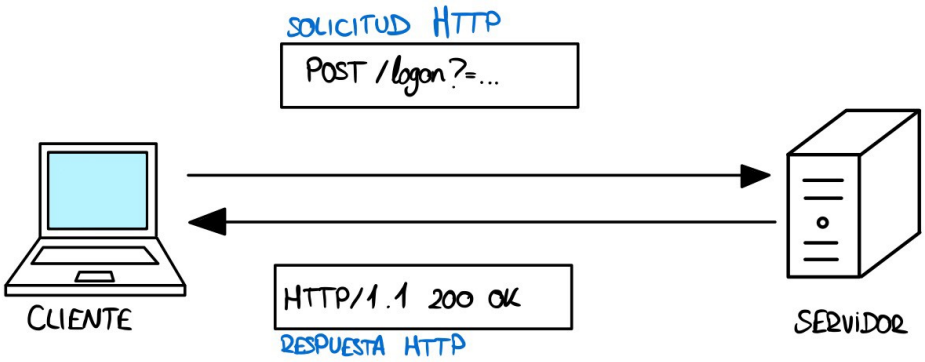

FIGURA 2-1: Ejemplo de transacción HTTP.

2.3 Protocolo DNS

Al igual que el protocolo HTTP, DNS (*Domain Name System* [3]) también pertenece a la capa de aplicación. DNS es una base de datos distribuida implementada en una jerarquía de servidores DNS y un protocolo de la capa de aplicación que permite a los *hosts* consultar precisamente, dicha base de datos distribuida.

DNS consta de dos programas: un programa que se ejecuta desde el lado del cliente (*host* que desea realizar una pregunta sobre una dirección) y otro que se ejecuta por parte del servidor (encargado de responder a la pregunta recibida con la dirección traducida).

Para entender el funcionamiento de este protocolo, se propone el siguiente ejemplo:

1. El cliente ejecuta desde su lado la aplicación DNS.
2. El navegador extrae el nombre del host del URL y, pasa el nombre del *host* al lado del cliente de la aplicación DNS.
3. El cliente DNS envía una consulta (que contiene el nombre del *host*) a un servidor DNS.
4. El cliente recibe una respuesta con la dirección IP correspondiente al nombre del *host*.
5. Una vez el navegador ha recibido la dirección IP de servidor DNS, se procede a iniciar una conexión TCP con el proceso servidor HTTP en dicha dirección.

En este TFG no se desarrollan aplicaciones DNS para realizar consultas. En cambio, sí que se realiza un análisis exhaustivo de los registros de flujos de red, incluyendo la carga útil a nivel de byte. Para ello, es necesario conocer a fondo la cabecera de este protocolo y cuál es la diferencia entre una pregunta y una respuesta. A continuación, se pueden ver los campos que componen la cabecera DNS:

- Los primeros 2 bytes, representan el campo de **identificación** de la transacción DNS.
- Los siguientes 2 bytes, contienen distintos ***flags*** (banderas) que proporcionan información acerca de la conexión establecida. En la FIGURA 2-2 se pueden ver todos los *flags* que contiene la cabecera DNS:

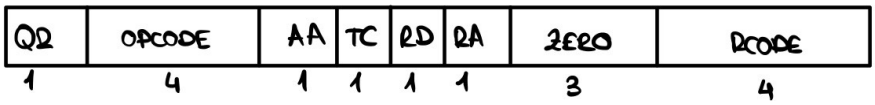

FIGURA 2-2: *Flags* DNS.

- **QR.** Campo de un 1 bit que indica si el mensaje es una pregunta o una respuesta (**0** significa **pregunta** y **1** significa **respuesta**).
- **OPCODE.** Campo de 4 bits que proporciona información acerca del mensaje; El valor estándar es 0 pero otros valores son: 1(pregunta inversa) y 2 (solicitud del estado del servidor).
- **AA.** Campo de 1 bit que confirma que el mensaje se trata de una respuesta autoritativa (de su traducción en inglés *Authoritative Answer*).
- **TC.** Campo de 1 bit que significa Truncado. Con UDP, ocurre si el tamaño total de la respuesta excedió el tamaño máximo (habitualmente 512 bytes, aunque se trata de un valor configurable como se puede ver en [17]), y sólo se devolvieron los primeros bytes de la respuesta.

- **RD.** Campo de 1 bit que afirma Recursión Deseada (de su traducción del inglés *Recursion Desired*). Este bit se puede establecer en una consulta y luego se devuelve en la respuesta. Esta bandera le dice al servidor de nombres que maneje la consulta en sí, llamada consulta recursiva.
- **RA.** Campo de 1 bit que indica que la recursión se encuentra disponible. Este bit se establece en 1 en la respuesta si el servidor admite la recursión.
- **ZERO.** Campo de 3 bits que debe de ser 0.
- **RCODE.** Campo de 4 bits que contiene el código devuelto. Los valores comunes son 0 (sin error) y 3 (error de nombre). Un error de nombre se devuelve sólo desde un servidor de nombres autorizado y significa que el nombre de dominio especificado en la consulta no existe.

- Tras los *flags*, se encuentra representado por otros 2 bytes el **número de preguntas**.
- Tras el número de preguntas, se encuentra el **número de respuestas** al cual se le adjudican 2 bytes igualmente.
- Después, aparece el **número de respuestas autoritativas**, representado también por 2 bytes.
- Finalmente, para concluir con el tamaño fijo de 12 bytes se tiene el campo de **respuestas adicionales** representado también por 2 bytes.

Esto, supone el fin del tamaño fijo del mensaje, y a continuación entran en juego los campos que contienen un número variable de bytes. Esto ha de ser tratado con mucho cuidado durante el desarrollo del proyecto, dado que se están leyendo datos de un archivo en el cual las transacciones DNS siguen esa cabecera.

Se debe destacar por la razón mencionada en el párrafo anterior cómo se organizan las preguntas y las respuestas, cosa que podemos ver en las siguientes dos imágenes:
En la FIGURA 2-3 se aprecia la estructura de las **Preguntas DNS**:

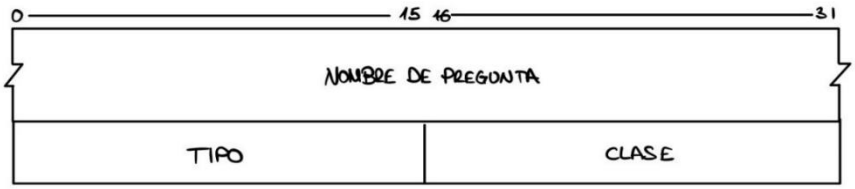

FIGURA 2-3: Estructura de una pregunta DNS.

En la figura FIGURA 2-4 se aprecia la estructura de las **Respuestas DNS**:

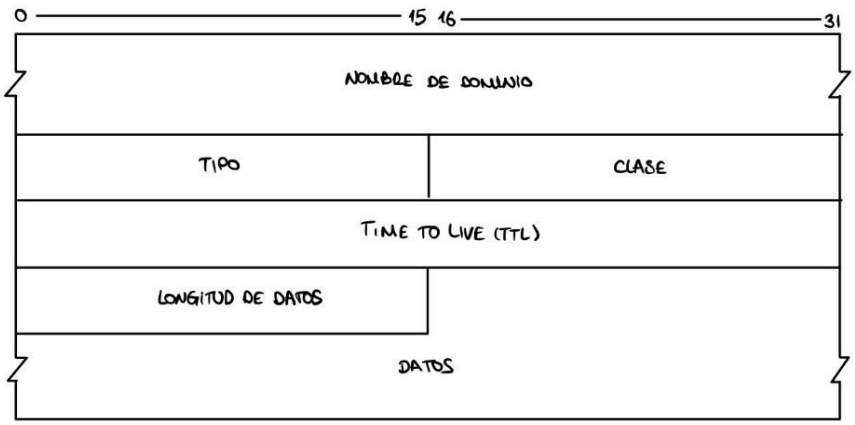

FIGURA 2-4: Estructura de una respuesta DNS.

Como se puede apreciar, existen una serie de diferencias que no pueden ser pasadas por alto. Ambos formatos, tienen en común el nombre de la pregunta o la respuesta, así como el tipo y la clase de pregunta o respuesta que ha sido realizada.

En particular, las **consultas** o **preguntas** son una secuencia de una o más etiquetas, en donde cada etiqueta comienza con un recuento de 1 byte que especifica el número de bytes que siguen. El nombre termina con un byte de 0, que es una etiqueta con una longitud de 0, que es la etiqueta de la raíz. Cada byte de conteo debe estar en el rango de 0 a 63, ya que las etiquetas están limitadas a 63 bytes. Se verá más adelante en esta sección que un byte de conteo con los dos bits de alto orden activados, valores 192 a 255, se utiliza con un esquema de compresión. El esquema de compresión se ha convertido en todo un reto durante el desarrollo de este proyecto porque supone realizar comparaciones a nivel de byte que hacen de la programación un trabajo laborioso. La FIGURA 2-5 indica cómo se almacena el nombre de dominio *www.uam.es* :

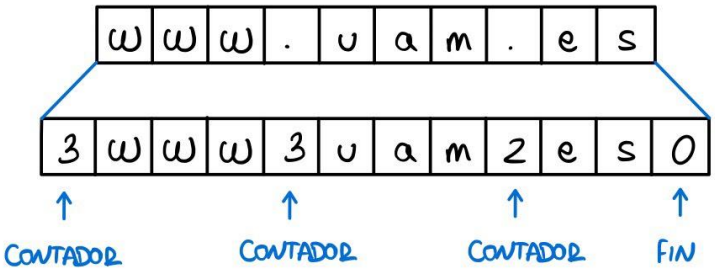

FIGURA 2-5: Nombres de dominio con formato DNS

Por su parte, las respuestas tienen la particularidad del campo TTL (*Time-to-Live*). Este campo nos indica el número de segundos que el RR (*Resource Record*) puede ser almacenado en caché por el cliente. Se pueden definir los Registros de Recursos DNS como entradas a la base de datos del Servidor DNS. Los registros de recurso suelen ser un mapeo de nombre a dirección IP (IPv4 o IPv6) o viceversa. Los registros de recursos DNS se utilizan para responder a las consultas de los clientes DNS.

La longitud de los datos de recurso, tal y como su nombre especifica, proporciona información sobre la cantidad de datos de recurso. El formato de estos datos depende del tipo. Para un tipo de 1 (un registro "A"), los datos del recurso son una dirección IP de 4 bytes como se puede ver en [4].

Durante el desarrollo de este TFG, manejar los campos de longitud variable ha sido todo un desafío. Sin la información proporcionada por la cabecera DNS, en efecto, hubiera sido completamente imposible el desarrollo de la herramienta *DNS Flow Analyzer*, dado que al trabajar sobre bytes y no sobre texto plano la información que se recibe sobre las consultas o las respuestas sería indescifrable.

2.4 Análisis de flujos de red

La monitorización de flujos ha ido ganando terreno en los últimos años. Es a principio de la década de 1990 cuando se crea el primer *IETF WG* (*Internet Engineering Task Force Work Group*), aunque es en 2002 cuando se implementa *Netflow v5* y realmente el análisis de flujos comienza a convertirse en una herramienta potente, así como actualizable como se expresa en [5]. A este método, no cabe duda de que se le añadirán funcionalidades adicionales en el futuro y se convertirá en un método puntero en cuanto al seguimiento de tráfico en las redes

se refiere. Centrándose en el análisis de flujos, se obtiene una visión de conjunto de las transacciones que circulan por la red, en lugar de si se analizan paquetes individualmente. Monitorizando flujos, se presenta la oportunidad de abarcar toda la cadena de observación de paquetes, utilizando herramientas de exportación de flujos y los protocolos correspondientes como *NetFlow* e IPFIX (*IP Flow Information Export*). En contraste con lo que esto supone frente al análisis de paquetes, todas las etapas de monitoreo de flujos están entrelazadas tal y como se puede notar en [5].

Cada una de las etapas mencionadas en [5], tiene que ser comprendida a fondo, antes de comenzar a realizar mediciones sobre los registros de flujos. De lo contrario, se corre riesgo potencial de pérdida de datos, que puede afectar a los resultados obtenidos.

2.4.1 Diversos analizadores de registros de flujos de red

Se realiza a continuación un pequeño análisis de las empresas que hoy en día destacan en análisis de registros de flujos, así como una pequeña comparación del trabajo y las herramientas que desarrollan estas empresas, frente al proyecto y las herramientas desarrolladas durante este TFG. Se ahondará tanto como sea posible, de tal forma que se pueda estudiar la información que estas empresas hacen pública, que en ocasiones es un poco limitada.

2.4.1.1 Flowmon

Flowmon es una empresa con sede en la República Checa. Indagando acerca de esta empresa, se ha encontrado en su propia página web una sección de verdadera utilidad. En ella se puede realizar una *demo* que simula el comportamiento de la herramienta. Se puede comprobar que es una herramienta muy completa que analiza tráfico, paquetes y flujos.
El caso que concierne a este proyecto es el de los flujos de red, por lo que, si se centra la atención en ellos como se ha hecho en la FIGURA 2-6:

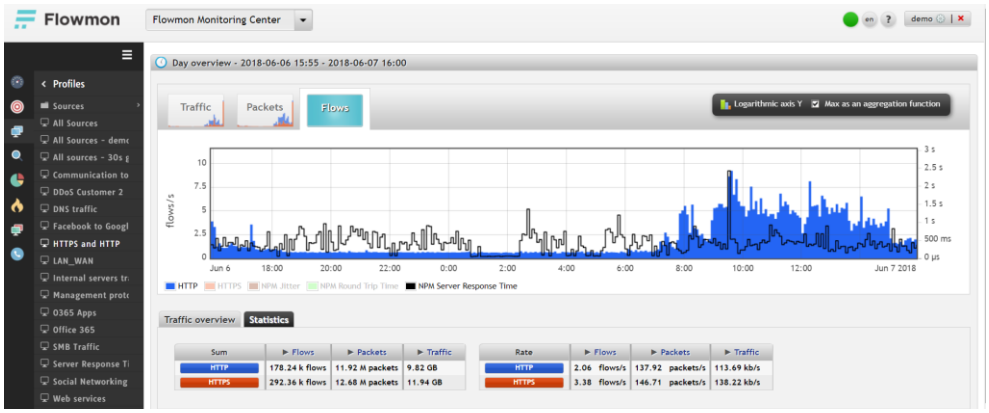

FIGURA 2-6: Demo de la herramienta de análisis Flowmon. Protocolo HTTP.

Se puede comprobar como *Flowmon* analiza el número de flujos HTTP, en la propia pantalla de "*Flows*". Aparte de realizar el análisis correspondiente la herramienta dispone de una interfaz gráfica muy completa.
En esta pantalla de estadísticas, sin llegar a saber si es porque únicamente se dispone de un acceso de nivel "demo", únicamente se muestra por pantalla el número de flujos o el número de flujos por segundo, paquetes y tráfico existentes a lo largo de la conexión. Esta información, en cuanto a estadísticas, carece de información útil si se compara con las

estadísticas que puede llegar a ofrecer el HTTP *Flow Analyzer* desarrollado en este trabajo, dado que es capaz de mostrar, por ejemplo, el tipo de consultas/respuestas HTTP que han sido realizadas durante la captura de tráfico, así como exponer los códigos de respuesta de las transacciones. Esto se puede comprobar según [6].
Lo que al tráfico DNS respecta, ha sido obtenido de nuevo con la versión web. Mediante la demo online la información que se puede ver se presenta en la FIGURA 2-7:

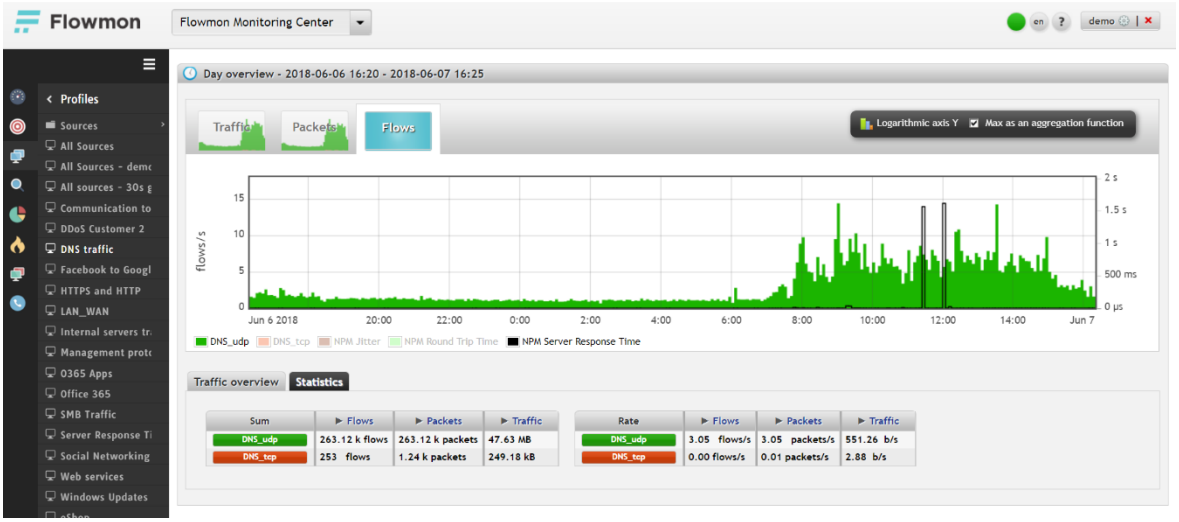

FIGURA 2-7: Demo de la herramienta de análisis Flowmon. Protocolo DNS.

Ocurre lo mismo que en el caso anterior. Desde el punto de vista de análisis de flujos, parece un poco escueto mostrar únicamente el número de flujos y número de flujos por segundo que ha aparecido durante la captura. El DNS *Flow Analyzer*, por su parte, es capaz de mostrar el tipo tanto de respuestas como de consultas, así como el número (en valor absoluto y porcentaje relativo al total) de preguntas/respuestas que fueron realizadas.

2.4.1.1 Solarwinds

Solarwinds se posiciona hoy en día como empresa puntera en cuanto al análisis de flujos de red se refiere. Disponen de multitud de herramientas verdaderamente potentes para llevar a cabo procesos de análisis y optimización de funcionamiento de red. En concreto, la herramienta que se dedica al análisis de flujos de red recibe el nombre de *NetFlow Traffic Analyzer*. Para poder estudiar más a fondo esta herramienta, se procedió a descargar una versión "*Free Trial*". Esta versión dispone de todas las características y funcionalidades hasta un máximo de 30 días según [7].

Sin embargo, durante la instalación, resalta un error que indica que no es posible completar este proceso. Se debe al problema que se muestra en la FIGURA 2-8:

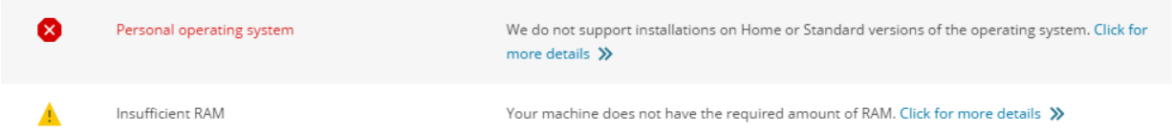

FIGURA 2-8: Error de instalación de *NetFlow Traffic Analyzer*.

Si se siguen las indicaciones para obtener más detalles sobre los problemas acontecidos, se puede observar como la memoria RAM (*Random Access Memory*) que la herramienta de *Solarwinds* necesita un mínimo de 16 GB. Dadas las limitaciones del equipo personal, este es un valor elevado y la instalación no se puede llevar a cabo. Esto, sumado a que no se pueden realizar instalaciones en entornos Windows con distribuciones Home o Personal hace imposible que se pueda llevar a cabo una evaluación de la herramienta.

2.4.1.2 Talaia

Talaia es un ejemplo de *Spin-off*, surgida en la UPC (Universidad Politécnica de Cataluña). Tiene clientes de carácter internacional y en concreto, su proyecto núcleo está orientado al estudio de flujos de red.

Para entender bien a que se dedican y poder realizar una comparación con las herramientas desarrolladas durante este proyecto, se ha llevado a cabo una prueba de la herramienta que Talaia ofrece. Esta prueba, está disponible en su página web referenciada en el enlace [18], permite realizar un análisis muy completo, no solo de flujos de red, sino que ofrece información sobre posibles anomalías, geolocalización del tráfico, direcciones sobre las que se realizan mayor número de transacciones, etc.

Si se realiza una comparación con HTTP *Flow Analyzer* o con DNS *Flow Analyzer*, se puede ver como la herramienta desarrollada por Talaia tiene mejores prestaciones ya que a la vez que estudia el tráfico que circula por la red es capaz de extraer las estadísticas correspondientes de todo lo analizado. Cabe destacar que no solo se dedica a estudiar protocolos HTTP y DNS, pero en lo que respecta a estos dos, ya que son objeto de estudio en este TFG, las estadísticas que Talaia ofrece, son más escuetas frente a los resultados finalmente mostrados en el capítulo de validación 5.2. En cuanto a HTTP respecta, ofrece estadísticas sobre los puertos sobre los que más transacciones se han realizado, las direcciones más visitadas y las direcciones que más transacciones han llevado a cabo entre sí. De igual forma, detallan información para el protocolo DNS. Se podría decir que tanto el HTTP *Flow Analyzer* como el DNS *Flow Analyzer* serían un buen complemento para el proyecto en general.

Después de haber realizado una prueba sobre la herramienta desarrollada por Talaia, parece extraño el hecho de que para ambos protocolos se ofrezca la misma información y no se muestren estadísticas sobre los métodos más utilizados en el caso HTTP o los tipos de preguntas/respuestas para el caso DNS. Como se presenta en [8] existe un estudio sobre las técnicas existentes para la monitorización de redes, pero, Talaia utiliza únicamente los registros de flujos de red para extraer conclusiones. Es por esto por lo que no ofrece estadísticas sobre los métodos utilizados, los códigos de respuesta, etc.

Finalmente, cabe destacar como se puede leer en [8] el hecho de que se utilice ML (*Machine Learning*), que optimiza el proceso para la herramienta de análisis. Esta característica podría implementarse como mejora en un futuro sobre las herramientas desarrolladas durante este proyecto.

En este TFG, se ha llevado a cabo la implementación de dos herramientas más sencillas que el conjunto que Talaia presenta. Esto hace que tanto el HTTP *Flow Analyzer* como el DNS *Flow Analyzer*, puedan llegar a formar parte de un proyecto que englobe a estas herramientas. Es por ello por lo que, para comparar las funcionalidades de las herramientas,

se ha centrado la atención únicamente en los protocolos HTTP y DNS. El desarrollo de este proyecto está basado en la herramienta desarrollada por la cátedra UAM Naudit, HTTP *Dissector*. Esto se ve detallado en [10].

2.5 M³OMON

M³OMON es una herramienta desarrollada por el grupo de investigación **HPCN** (*High Performance Computing and Networking*) de la Universidad Autónoma de Madrid descrito en [9].

Se trata de una arquitectura de sistema que ayuda a desarrollar aplicaciones de monitorización y a realizar diagnósticos de red. Se posiciona como intermediario entre el tráfico y las aplicaciones de monitorización, proporcionando características avanzadas combinando alto rendimiento y bajo coste.

M³OMON se aprovecha de un enfoque multigranular y multipropósito para atajar el problema de la monitorización en las redes. La monitorización multigranular da respuesta a tareas que utilizan agregados de tráfico para identificar ciertos eventos en la red, y requiere registros de flujo o de datos de paquetes. En numerosas ocasiones se requieren ambos para entenderlo y tomar medidas.

Por ello, esta herramienta proporciona estadísticas a nivel de paquete, a nivel de flujo y agregadas. Es entonces cuando el diseño multipropósito permite **no** solo realizar las tareas en paralelo que están dirigidas a diferentes propósitos relacionados con el tráfico, sino también a compartir granularidades entre aplicaciones. Por ejemplo, varias aplicaciones alimentadas a partir de flujos proporcionados por M³OMON.

El bajo coste viene dado por la combinación de software libre y hardware de propósito general, mientras que el alto rendimiento se obtiene gracias a las modificaciones del controlador NIC (*Network Interface Controller*) estándar, la gestión eficiente de memoria y la optimización en cuanto temas de programación.

En la mayoría de los casos, realizar un buen diagnóstico de red y poder solucionar los posibles problemas existentes, implica seguir los siguientes pasos:

1. Los agregados de datos se utilizan para detectar una situación anómala.
2. Las trazas de nivel de flujo se emplean para identificar los agentes problemáticos.
3. La inspección mediante rastreo de tráfico finalmente se realiza con la intención de diagnosticar y dar solución al problema.

Es entonces cuando la granularidad, se convierte en multigranularidad para M³OMON. Se descubre que esta filosofía proporciona a los profesionales una capa intermedia de software capaz de proporcionar acceso a datos multigranulares a través de una API unificada.

Por lo tanto, desde el punto de vista de un profesional, no hay diferencia entre pedir un paquete o un registro de flujo. M³OMON define en su versión actual tres niveles de granularidad, es decir, agregados de series temporales (por ejemplo, series similares a MRTG (*Multi Router Traffic Grapher*)), registros de flujo y trazas de paquetes.

Todas estas granularidades pueden ser recuperadas en tiempo real o después de haber sido almacenadas en un disco duro. Es aquí donde M^{3}OMON realmente destaca, dado que esto supone una clara desviación del estado actual de la técnica que se ha centrado en la captura, el almacenamiento y la creación de registros de flujos de paquetes como procesos separados.

2.5.1 HPCAP y DetectPro

De la mano de M^{3}OMON nacen estas dos herramientas, **HPCAP** y **DetectPro** descritas en [9], destinadas a complementar y a optimizar el desarrollo seguido hasta ahora. Se hará hincapié en esta sección en *DetectPro* dada su relación con este TFG. En este proyecto, se toman los registros provenientes de *DetectPro* así como las cargas útiles de los flujos para obtener resultados sobre ellos, por lo tanto, de no ser por *DetectPro* hubiera sido imposible llevar a cabo el desarrollo de este trabajo.

DetectPro es una herramienta complementaria implementada sobre el marco de trabajo propuesto y, es capaz de proporcionar estadísticas de seguimiento, informar de las alarmas y, posteriormente, realizar análisis forenses basados en el nivel de paquetes, trazas, registros de nivel de flujo y registros estadísticos agregados.

DetectPro lee estadísticas agregadas para diagnosticar cambios a corto y largo plazo, e informa de las alarmas correspondientes, incluyendo información de las anomalías detectadas como el inicio y el final de tiempos de la alarma, así como los valores del bit/paquete/flujo que causaron la alarma. Cuando una alarma se dispara, *DetectPro* inicia automáticamente la inspección de los registros para extraer información sobre la actividad de la red durante el periodo de alarma. Es entonces cuando genera varios informes que contienen, por ejemplo, la distribución de hosts/segmentos de red/servicios con el mayor número de bytes, paquetes y flujos, así como otras métricas (por ejemplo, banderas TCP, retransmisiones), y los mayores flujos en términos de paquetes y/o bytes.

2.6 Conclusión

Tras conocer las bases que asientan este proyecto, se puede ver como el análisis de registros de flujos, proporciona información muy útil a los profesionales que tratan de conocer qué eventos pueden estar afectando a las redes de sus propios comercios o empresas. Además, se debe resaltar que realizar análisis sobre registros de flujos no es lo mismo que realizar los análisis paquete por paquete, pero sí se puede extraer información completa sobre anomalías o eventos existentes en las transacciones de red.
Finalmente, se han dado a conocer una serie de herramientas que permiten verificar como el uso de flujos de red para la monitorización es un hecho en el presente. Prestando especial atención a las herramientas desarrolladas por la cátedra UAM Naudit. Estas son las que han permitido mediante los flujos y los registros provenientes de *DetectPro* que el desarrollo que en los siguientes capítulos se detalla sea posible.

3 Diseño

3.1 Introducción

En este capítulo se describe el proceso previo a realizar el desarrollo de las herramientas de análisis. Se destaca la relación que tienen los archivos con los cuales se ha trabajado, de tal forma que se entienda como se puede proceder a analizarlos. Se detallará, la estructura de diseño que tienen ambas herramientas pensadas para analizar los registros de flujos de red contenidos en los archivos recibidos.

3.2 Lenguaje de programación

Tanto para el desarrollo del HTTP *Flow Analyzer* como para el DNS *Flow Analyzer*, el lenguaje de programación escogido es C. Esto se debe a que ofrece un alto rendimiento permitiendo utilizar características de bajo nivel para poder realizar una implementación óptima y eficiente.

Este lenguaje fue escogido respecto de otros como pudo ser Python o Java porque, además de permitir que se llevaran a cabo operaciones de bajo nivel, es un lenguaje que presenta características clave para una gestión optimizada del uso de memoria por parte de las herramientas.

Adicionalmente, C es un lenguaje con el cual ha sido frecuente el trabajo durante el desarrollo del grado. Lo que permite partir de una base más sólida que si se comenzara desde un lenguaje completamente nuevo.

3.3 Estudio de los registros de flujos de red

El tráfico en una red de datos puede considerarse como un flujo que pasa a través de los elementos de la red. Para fines administrativos o de otro tipo, a menudo es interesante, útil o incluso necesario tener acceso a la información sobre estos flujos que pasan a través de los elementos de la red. Un Proceso de Recolección debe ser capaz de recibir la información de Flujo pasando a través de múltiples elementos de red dentro de la red de datos. Esto requiere uniformidad en el método de representación de la información de los flujos y en los medios de comunicación de los flujos desde los elementos de la red hasta el punto de recogida.

Los flujos de red ofrecen, por tanto, la posibilidad de extraer estadísticas acerca del comportamiento de una red de carácter verdaderamente útil. Para el caso particular de este proyecto, en cuanto a HTTP se refiere, se han extraído campos como el Host, URL, códigos de respuesta, métodos, etc. En el caso del protocolo DNS se ha podido comprobar cómo se podía extraer información acerca de banderas relevantes como el número de consultas/respuestas, así como la pregunta realizada y las respuestas recibidas por el servidor de DNS, el identificador de la transacción DNS, etc.

Para comenzar a diseñar las herramientas, adquiere una importancia vital el hecho de saber cómo están estructurados los datos. Primeramente, se encuentra un archivo escrito en texto plano (archivo1.txt) en el cual se presentan una serie de campos que proporcionan información sobre el flujo que han tenido las transacciones llevadas a cabo el día en el que se hizo la captura de tráfico. En segundo lugar, se encuentra un archivo binario

(archivo2.bin) en el cual, se tiene el contenido de las transacciones llevadas a cabo. En este archivo es en el cual aparece la descripción de las conversaciones entre cliente y servidor para los protocolos estudiados.

3.3.1 Estructura de los registros: archivo de texto plano

Para diseñar de forma eficiente las herramientas, se debe tener en cuenta cómo están estructurados los registros que se utilizaron. El separador utilizado para diferenciar cada campo es un espacio en blanco y entre los campos existentes, se deben destacar los siguientes dada su importancia en este proyecto:

1. IP origen: dirección IP desde la cual se origina el flujo.
2. IP destino: dirección IP a la cual está destinado el flujo.
5. Protocolo de transporte: representado por una serie de caracteres. Un ejemplo del formato en el que está representado es el siguiente: p6 (letra minúscula "p" seguida del número de protocolo correspondiente). Esta notación se adopta conforme a la asignación de números a los protocolos de IANA (*Internet Assigned Numbers Authority*) detallada en [13].
6. Puerto origen: Número de puerto desde el cual se origina el flujo.
7. Puerto destino: Número de puerto al cual está destinado el flujo.
10. TS inicio: *Timestamp* en el que se inicia la transacción.
11. TS fin: *Timestamp* de final de la transacción.
31-32 Nº de bytes de *payload* y *offset* en el archivo de *payload* (si lo hubiera): estos dos campos se desarrollan a continuación debido a su impacto sobre el análisis.

Tras haber visto la estructura de este archivo, antes de pasar a estudiar el archivo binario, se pensó en diseñar un análisis de estos campos separados con espacios mediante awk. Fue entonces cuando aparte de analizar los campos, se pudo comprobar la importancia de los campos 31 y 32 (número de bytes de *payload* y número de bytes de *offset* respectivamente) sobre el archivo binario.

3.3.2 Estructura de los registros: archivo binario

Para el diseño de las herramientas, es difícil describir la estructura del archivo binario dada su complejidad para ser leído como se está acostumbrado. Cuando se habla de complejidad se aclara el nivel con un ejemplo:

,×D±öœ4Eè~ž‡Áéð"õwyÆ¹BýBõŒ‹y•çè4ÚŠý–¼¨d÷v”Š§÷-p“ÅŒ

Por lo tanto, carece de sentido ni si quiera intentar definir una estructura para este tipo de archivos. En cambio, sí que es importante conocer el comportamiento de este tipo de archivos frente al lenguaje de programación para el diseño de cada herramienta.

En el caso que afecta al tráfico HTTP, se dispone de información bastante más clara dado que la conexión y la conversación con el servidor queda registrada en formato de texto plano. Fue en este punto cuando se tuvo que pensar en el diseño del HTTP *Flow Analyzer*, lo que significó en este caso llevar a cabo un estudio sobre expresiones regulares en C. Gracias a este diseño, se consigue un formato de herramienta generalizado que aborda el problema planteado por un elemento físico de la red, los servidores. Casi en la totalidad de las ocasiones, no se puede establecer la forma en la que el servidor genera una respuesta indicando, por ejemplo, el host de la transacción o la URL de la conexión HTTP.

Por otro lado, se encuentra el análisis de los registros de flujo de red con respecto al protocolo DNS. En esta ocasión hay que rediseñar el proceso mediante el cual se analiza el archivo binario porque la lectura de los datos que se analizaron no es directa, esto quiere decir que, en esta ocasión no se analiza texto plano, sino que se lleva a cabo un análisis a bajo nivel de los bytes que componen este archivo. Supuso todo un reto el hecho de, además de tener en cuenta la composición de la cabecera DNS, que los mensajes (sobre todo de respuesta) tuvieran un formato de compresión, esto sumado a los campos de longitud variable de la cabecera DNS hacía que, la lectura por bytes del archivo binario tuviera que tener un control sobre el número de bytes que se habían leído.

3.3.3 Relación entre archivo binario y el archivo de texto plano.

Tal y como se destacaba antes, existen ciertos campos del archivo de texto plano que juegan un papel fundamental en la comprensión de los flujos. Los campos que contienen el número de **bytes de *payload*** y el número de **bytes de *offset*** realizan una función similar a un marcador de página, como los que se usan a diario para no perder la guía en un libro que se esté leyendo.

En este proyecto, no se lee un libro sino un archivo binario y uno de texto en paralelo. Para la lectura del archivo binario, número de bytes de *offset*, dado por el archivo de texto plano, determina la posición desde la cual se comienza a leer y el número de bytes de *payload*, también dado por el archivo de texto plano, indica el número de bytes que van a ser leídos desde el *offset* establecido. Para comprender mejor el enlace entre ambos se presenta la FIGURA 3-1:

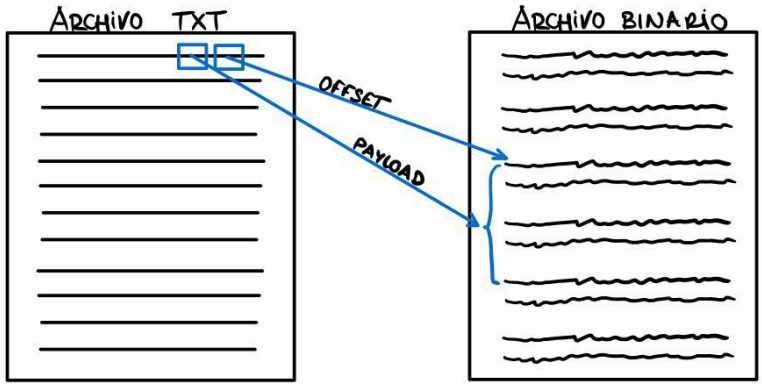

FIGURA 3-1: Relación entre los archivos de texto plano y binario

3.4 Estructura de HTTP Flow Analyzer

Para diseñar el HTTP *Flow Analyzer*, se llevó a cabo una descomposición de la cuestión para que esta, no desbordara en ningún momento y para poder tener claros en todo momento los objetivos que se plantean en este TFG. Primero, se pensó en analizar los registros de flujos de red mediante los puertos. Los puertos más comunes para las transacciones HTTP son el puerto 80 y 8080. Esto era sencillo dado que estos puertos aparecían en el archivo de texto plano y mediante métodos de *parseo* es fácil capturar el campo 6 y el campo 7 que contienen

respectivamente puertos de origen y destino. Este diseño proporciona alguna ventaja interesante como puede ser la precisión a la hora de analizar flujos HTTP, dado que como el tráfico que captura por estos puertos es HTTP se podía confirmar que la precisión obtenida sería muy alta.

En segundo lugar, se procedió a abrir los dos archivos en paralelo para para analizar a nivel profundo la información contenida en los flujos de red. Para ello, se pensó en diseñar una segunda versión del programa que analizara mediante expresiones regulares en C el contenido del archivo binario, este paso intermedio supuso un avance notable en la persecución de los objetivos debido a que, se podía verificar que era posible leer las partes que contenían información sobre transacciones HTTP y diseccionar dicha información para poder exportarla en un futuro. Esto se detalla en la FIGURA 3-2:

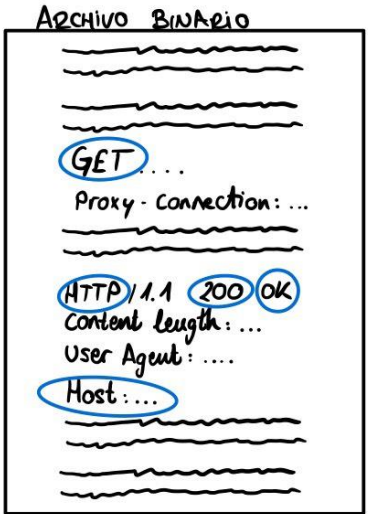

FIGURA 3-2: Disección del archivo binario

Es en una tercera etapa del diseño cuando se elimina el filtro de búsqueda por puerto. Esto, se hace gracias al estudio, como siempre, de los dos archivos conjuntos. El archivo de texto plano, por su parte, proporciona el protocolo por el cual se está llevando a cabo la transacción HTTP y, dentro del archivo binario se encuentra la información sobre esa transacción y el HTTP *Flow Analyzer* se diseña con el fin de que el filtro para aceptar que una transacción sea o no HTTP se haga estudiando si en algún sitio del archivo binario aparece una expresión HTTP como GET, POST, HTTP/1.1, etc.

El diseño final del HTTP *Flow Analyzer* es capaz finalmente de exportar la información que requiere de los registros de flujos de red como puede ser: host, URL, IP de origen/destino, puerto de origen/destino, etc. A continuación, en la FIGURA 3-3 se puede apreciar el diseño final del HTTP *Flow Analyzer*:

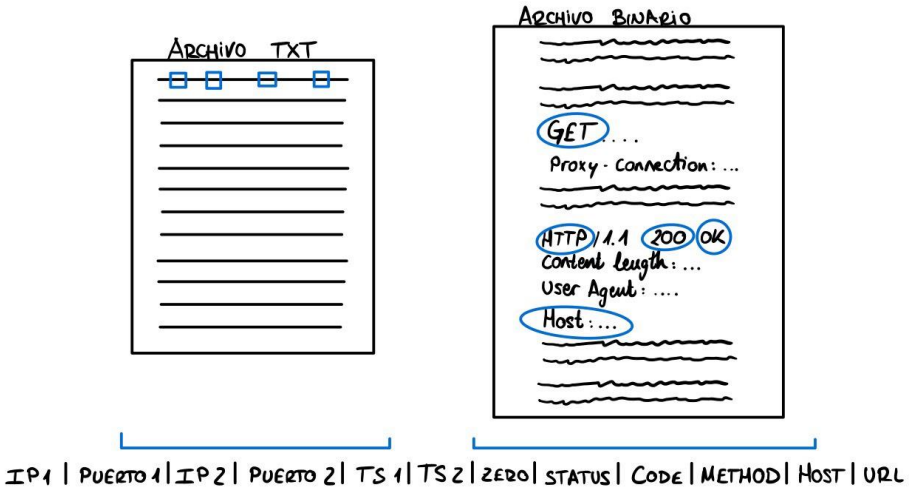

FIGURA 3-3: Diseño final: HTTP *Flow Analyzer*

3.5 Estructura de DNS Flow Analyzer

Para el diseño del DNS *Flow Analyzer*, hubo que tener en cuenta el hecho de que la información existente, al contrario que antes, no se dispone en texto plano sino en binario. Por esta razón, primero se diseñó una primera versión de la herramienta que leía los archivos de texto plano y binario, filtrando por puerto 53 y protocolo 17 (UDP) que son los parámetros habituales para las transacciones DNS. Esta versión estaba destinada a la toma de contacto con las cabeceras DNS y el trabajo de programación a nivel de byte. Esta simple versión se diseñó pensando en realizar un primer análisis que extrajera información sobre los campos de la cabecera con longitud fija.

Posteriormente, se diseña una segunda versión algo más completa que aparte de analizar los campos con longitud fija se dedicaba a analizar los campos de longitud variable como podían ser la pregunta DNS realizada o la respuesta recibida. En este aspecto, esta actualización del diseño suponía un reto debido a que, en multitud de ocasiones el hecho de hacer una pregunta DNS no implica recibir una única respuesta. Por estas razones, este segundo diseño conlleva una función adicional a la ya comentada, esta es: llevar la cuenta de bytes que han sido leídos para conocer el número de bytes que queda por leer.

Aunque siendo estrictos, el segundo diseño necesitaría de un posterior tercer diseño debido a que las transacciones DNS pueden llevar compresión. Esto, significa que dentro de un mensaje DNS existe la posibilidad de optimizar el espacio que este ocupa referenciando nombres que ya se tienen dentro de la transacción. A nivel de programación, supuso tener que llevar a cabo una tercera versión del programa que tuviera esto cuenta porque enc aso contrario, se empieza a acarrear un error en la lectura de los bytes provocado a partir de la compresión que concluye un claro fallo en la herramienta.
Para comprender mejor la compresión dentro del protocolo DNS se muestran la FIGURA 3-4 y la FIGURA 3-5, en donde primero se ve como se ha realizado una pregunta y para la respuesta se ha utilizado compresión:

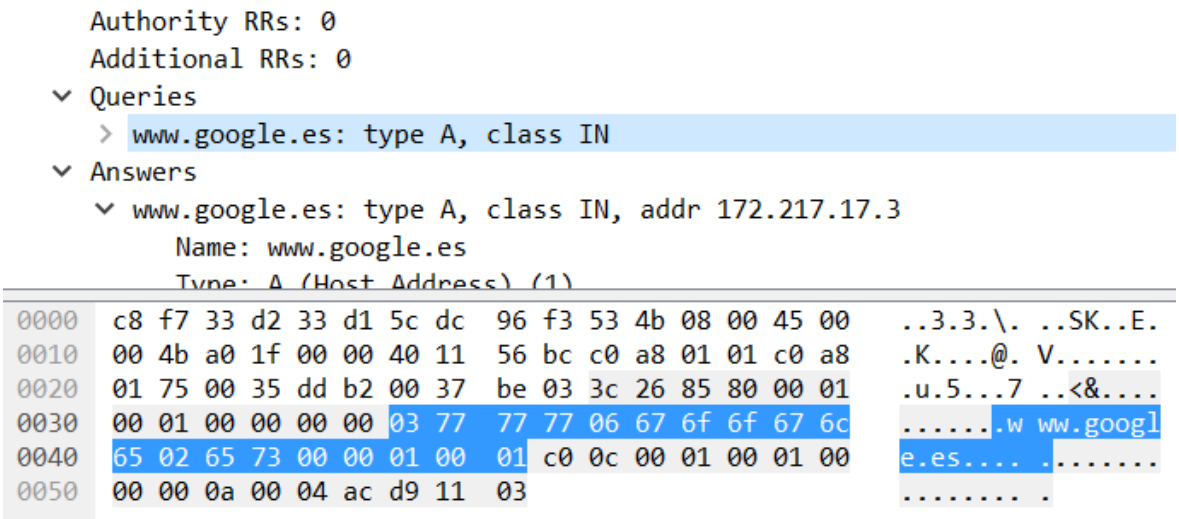

FIGURA 3-4: Ejemplo I: Compresión DNS

```
Authority RRs: 0
Additional RRs: 0
▼ Queries
  ▶ www.google.es: type A, class IN
▼ Answers
  ▼ www.google.es: type A, class IN, addr 172.217.17.3
      Name: www.google.es
      Type: A (Host Address) (1)
```

```
0000  c8 f7 33 d2 33 d1 5c dc  96 f3 53 4b 08 00 45 00   ..3.3.\. ..SK..E.
0010  00 4b a0 1f 00 00 40 11  56 bc c0 a8 01 01 c0 a8   .K....@. V.......
0020  01 75 00 35 dd b2 00 37  be 03 3c 26 85 80 00 01   .u.5...7 ..<&....
0030  00 01 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   .......w ww.googl
0040  65 02 65 73 00 00 01 00  01 c0 0c 00 01 00 01 00   e.es.... ........
0050  00 00 0a 00 04 ac d9 11  03                        ........ .
```

FIGURA 3-5: Ejemplo II: Compresión DNS

Después de los 3 diseños anteriores, se llega al diseño final. El cual, es capaz de dar cohesión a las características mencionadas anteriormente y analizar los registros de flujos de red de tal forma que proporciona información como: identificador de la transacción DNS, las preguntas y respuestas que se han llevado a cabo, IP de origen/destino, número de preguntas y respuestas vistas, etc. Se puede ver en la FIGURA 3-6 el diseño final del DNS *Flow Analyzer*:

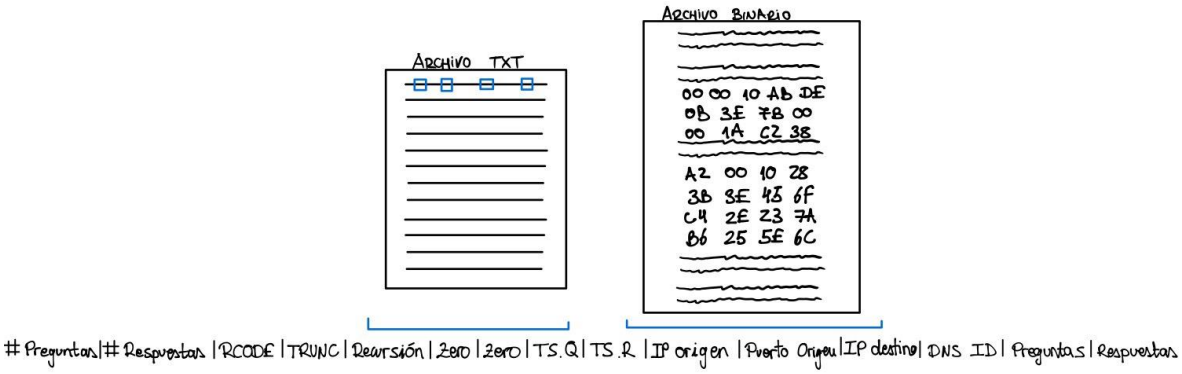

FIGURA 3-6: Diseño final: DNS *Flow Analyzer*

3.6 Conclusión

Tras completar el diseño de las herramientas, se tuvieron más claros los objetivos planteados al principio y se adquirió una visión periférica del funcionamiento al que se aspiraba llegar con estas herramientas. Se consiguió definir una serie de objetivos adicionales que derivaban del objetivo principal para poder tener el proyecto bajo control en todo momento y por último se aprendió el método de trabajo para realizar análisis de datos a bajo nivel.

4 Desarrollo

4.1 Introducción

En esta sección se lleva a cabo la presentación de las herramientas desarrolladas y se plantea un nuevo y fundamental concepto, registros de flujos de red simétricos. Se plantea el desarrollo de las herramientas viendo en ambas ocasiones cómo se lleva a cabo la disección de los flujos de red durante las distintas fases del proyecto y cómo se ha decidido segmentar los datos para una clara exportación de estos.

4.2 Registros de flujos de red simétricos

Este concepto toma nombre durante el desarrollo del HTTP *Flow Analyzer* pero esto no significa que no se utilice en el desarrollo del DNS *Flow Analyzer*. Por ello, se presenta a nivel general en el desarrollo del proyecto.

Los flujos como se explicó en la sección de diseño [3.3], están compuestos por varios campos. Para definir el concepto de flujo simétrico se utilizan los siguientes campos en particular:

- Dirección IP de origen.
- Dirección IP de destino.
- Puerto de origen.
- Puerto de destino.

Establezcamos un ejemplo para detallar este concepto:

1. Un equipo envía una petición, ya sea DNS, HTTP o cualquier otro protocolo, aunque no se haya estudiado en este proyecto. Esta petición genera un flujo que llega al destino correspondiente marcado por la dirección de destino.
2. El sistema que recibe esa petición genera un flujo de respuesta que contiene de dirección y puerto de destino, las direcciones del equipo que generó la petición.
3. Cuando se genera la respuesta y llega al equipo originario de nuevo, se considera que ha habido una conversación y a los dos flujos correspondientes a esta transacción los podemos llamar **flujos simétricos**.

La FIGURA 4-1 que se presenta a la derecha ayuda a detallar este concepto.

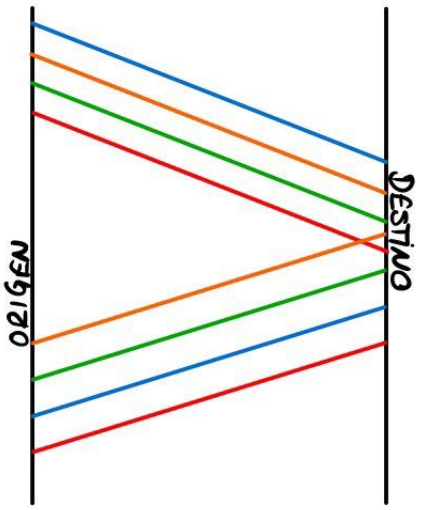

FIGURA 4-1: Representación de flujos simétricos

Como se puede apreciar en la imagen, la conversación entre cliente y servidor no tiene por qué estar ordenada temporalmente. Es por esto por lo que, cuando se emparejan dos flujos simétricos se utilizan los campos mencionados anteriormente.

¿Por qué entonces es tan importante este concepto? En cuanto al análisis se refiere, tanto la pregunta realizada como la respuesta tienen información clave para extraer. La única forma de hacer que esta información sea lo más completa posible son los flujos simétricos. Debido a que se obtiene información tanto de la petición como de la respuesta recibida.

4.3 HTTP Flow Analyzer.

Para el desarrollo del HTTP *Flow Analyzer* se sigue una estrategia de trabajo acorde al diseño de esta herramienta. Esto es, dividir la tarea de desarrollar esta herramienta es subtareas que completen el proceso.

Se desarrolla esta herramienta bajo el lenguaje de programación C. Con ello, se pretende que tanto memoria como otros recursos computacionales puedan ser aprovechados al máximo de forma eficiente.

Primero, se realiza una preparación de los datos que se van a llevar a cabo y para ello se recurre a la función *fopen*. Con esta función se realiza la apertura de hasta 3 archivos en paralelo:

- Archivo de texto plano.
- Archivo binario.
- Archivo de exportación final que contendrá la salida de la herramienta.

Se declaran también una serie de variables que permiten dar nombre a cada uno de los campos que contiene el archivo de texto plano. Posteriormente, se procede a leer línea por línea el archivo de texto plano para extraer la información necesaria que nos permita leer secuencialmente este archivo y el binario. Mediante una instrucción *while* y la función, otorgada por las librerías de C, *getline* se consigue leer línea a línea el archivo de texto plano hasta que *getline* devuelve valor -1 que significa que se ha alcanzado el final del archivo.

4.3.1 Disección de los registros de flujos de red.

Una vez preparados los datos, dentro del bucle que lee el archivo de texto, se lleva a cabo un análisis de cada una de las líneas del archivo de texto plano, mediante la instrucción *sscanf*. Esta instrucción nos permite analizar la línea extraída por *getline*. Es aquí cuando se averigua el valor de los campos que contienen la clave para realizar la lectura secuencial del archivo binario: el número de bytes de *payload* y el número de bytes de *offset*.
Antes de comenzar a leer el archivo binario se realiza un primer filtro encargado de verificar que el protocolo de transporte que aparece en cada momento es que se desea estudiar, en este caso el protocolo de transporte deseado es el p6, que significa TCP. (Se detallan en el anexo A el resto de números de protocolos existentes)

Una vez conocido el valor de estos campos, se procede a reservar memoria de forma dinámica para un buffer que almacenará la información que hay desde el número de bytes de *offset* hasta el número de bytes de *payload*. Mediante la función *fseek*, proporcionada por las librerías de C, se puede realizar la lectura secuencial del archivo posicionándose en un punto en concreto del archivo binario gracias al número de bytes de *offset*. Cuando se conoce que la posición en el archivo binario es la deseada, se recurre a la función *fread* que desde la posición establecida por *fseek* lee hasta el número de bytes de *payload* extraído. Además, se declara una variable extra para verificar, precisamente una vez eliminado el filtro de puertos, que el flujo bajo estudio tenga carácter HTTP.

4.3.2 Segmentación de los registros: expresiones regulares

Cuando se descubre que alcanzar el campo *Host* no es trivial, se recurre a algo parecido a las expresiones regulares en C. Las expresiones regulares se tratan de secuencias de instrucciones que contienen información para establecer una norma de carácter general sobre el funcionamiento de un programa.

Para el HTTP *Flow Analyzer* las expresiones regulares se utilizan a la hora de querer hallar el host de la transacción. Esto se lleva a cabo tal y como se puede ver en la siguiente línea de código:

```
sscanf(aux_buffer, "%s %s %*[^\n] %*[^H]Host: %s", aux_ReqType, aux_URL, aux_host);
```

De esta manera, se consigue analizar parte de la información almacenada en el buffer, la cual contiene los campos que en el desarrollo del HTTP *Flow Analyzer* nos conciernen.

Para comprender algo mejor el funcionamiento de las expresiones regulares se procede a explicar los ítems utilizados para este proyecto, se ha obtenido el siguiente ejemplo de [15]:

* El asterisco indica cero o más ocurrencias del elemento precedente. Por ejemplo, ab*c coincide con "ac", "abc", "abbc", "abbbc", etc.
% Una expresion compuesta por corchetes hacce que coincida con un solo carácter que se encuentra entre paréntesis. Por ejemplo, [abc] coincide con "a", "b" o "c".
^ Ajusta la posición inicial dentro de la cadena.

Gracias a la unión de la función *sscanf* junto a la sentencia de expresiones regulares, se obtiene la información deseada gracias a que las transacciones HTTP quedan registras en texto plano en el archivo binario.

Durante el desarrollo de esta herramienta se debía tener en cuenta el hecho de que exportar finalmente los datos obtenidos se haría acorde al programa descrito en el estado del arte [HPCAP y DetectPro], parte del proyecto M^{3}OMON: HTTP *Dissector*. Para posteriormente realizar validaciones y pruebas que acreditaran el trabajo realizado. Finalmente, a la hora de exportar los datos, la función elegida fue *fprintf* mediante la cual, se pueden imprimir los resultados obtenidos en otro archivo elegido como destino.

4.4 DNS Flow Analyzer.

Para el desarrollo del DNS *Flow Analyzer* se sigue una estrategia de trabajo acorde al diseño de esta herramienta. Es decir, dividir la tarea de desarrollar esta herramienta es subtareas que completen el proceso. Dado que el proceso inicial, es decir, el que tiene que ver con el archivo texto plano no cambia, en esta sección se asume que para la preparación de datos se ha seguido la misma filosofía.

Se debe destacar un desarrollo previo a la etapa de programación, durante el cual, se estuvo analizando a fondo la cabecera DNS, llegándose a realizar ejemplos de mensajes DNS para traducirlos a información legible mediante papel y bolígrafo, esto permitió comprender la cabecera DNS, así como el funcionamiento de sus campos variables.

4.4.1 Disección de los registros de flujos de red.

Una vez preparados los datos, dentro del bucle que lee el archivo de texto, se lleva a cabo un análisis de cada una de las líneas del archivo de texto plano, mediante la instrucción *sscanf*. Esta instrucción nos permite analizar la línea extraída por *getline*. Es aquí cuando se averigua el valor de los campos que contienen la clave para realizar la lectura secuencial del archivo binario: el número de bytes de *payload* y el número de bytes de *offset*.
Antes de comenzar a leer el archivo binario se realiza un primer filtro encargado de verificar que el protocolo de transporte que aparece en cada momento es que se desea estudiar, en este caso el protocolo de transporte deseado es el p17, que significa UDP.

Una vez conocido el valor de estos campos, se procede a reservar memoria de forma dinámica para un buffer que almacenará la información que hay desde el número de bytes de *offset* hasta el número de bytes de *payload*. Mediante la función *fseek*, proporcionada por las librerías de C, se puede realizar la lectura secuencial del archivo dado posicionándose en un punto en concreto del archivo binario gracias al número de bytes de *offset*. Cuando se conoce que la posición en el archivo binario es la deseada, se recurre a la función *fread* que desde la posición establecida por *fseek* lee hasta el número de bytes de *payload* extraído. Además, se declara una variable extra para verificar, precisamente una vez eliminado el filtro de puertos, que el flujo bajo estudio tenga carácter HTTP.

4.4.2 Segmentación de los registros: procesamiento de bytes

Una vez han sido diseccionados los datos, se lleva a cabo una llamada a la función *parseDNS* que se encarga de analizar toda la cabecera DNS campo a campo. Para analizar la cabecera DNS se llevan a cabo una serie de llamadas recurrentes a la función *memcpy*, otorgada por las librerías de C. Con esta función se analizan los distintos valores de los campos de la cabecera y se guardan en la variable que corresponda, esto se puede comprobar en las siguientes líneas de código:

```
memcpy(&aux16b, datagram+sizeRead, sizeof(uint16_t));
auxId=ntohs(aux16b);
sizeRead+=sizeof(uint16_t);
```

En la sentencia anterior, se tiene una primera variable denominada *aux16b* que realiza la función de variable auxiliar para ir almacenando la información leída. La variable *datagram* es en la que esta almacenado el array de bytes que contiene la respuesta/pregunta DNS y para este ejemplo *auxID* es la variable en la que estará almacenado el identificador de la transacción DNS. Esta sentencia se repite para el resto de los campos de forma que se va almacenando el número de bytes leído en la variable *sizeRead*. Esta variable, hace que no se pierda esa cuenta de bytes dado que como se ha comentado anteriormente la existencia de campos variables podría suponer un problema si no se lleva la cuenta de bytes leídos.

Posteriormente, cuando se llega al primer campo variable (nombre de la consulta DNS) se recurre a una segunda función: *get_DNS_name*. Esta función es la encargada de devolver el nombre en formato DNS realizando reconstrucciones siguiendo los punteros DNS que se han ido guardando. Almacena en una variable denominada: *reconstructedName* el nombre reconstruido y devuelve el número de caracteres que se han leído hasta encontrar el primer puntero DNS en caso de que lo haya o hasta encontrar el final de la cadena DNS. Es en esta función cuando se presta atención a la compresión que aparece en los mensajes DNS dado que también se utiliza para reconstruir el campo de las respuestas DNS que tengan formato texto.

Gracias a que esta función devuelve el número de caracteres, se puede seguir llevando la cuenta de los bytes leídos incrementando la variable *sizeRead* con el valor devuelto por la función *get_DNS_name*. Se realiza un bucle *for* que itera sobre el mensaje DNS por si hubiera más de una pregunta y, como el campo que contiene el número de preguntas ya ha sido analizado se tiene el dato para realizar el número de iteraciones correcto. Esto ocurre de igual manera en las respuestas, se lleva a cabo un *for* que itera hasta llegar al número de preguntas marcado por el campo de la cabecera que contiene el número de respuestas.

Para el caso de las respuestas, una vez se ve el nombre, el tipo y la clase que estas tienen, se procede analizarlas en función al tipo debido a que la forma de analizar una respuesta que contenga una cadena de caracteres no es lo mismo a analizar una respuesta que contenga una dirección IPv4 o IPv6.

Para aclarar estas diferencias se presentan las siguientes líneas de código en las que se puede apreciar la diferencia de análisis entre un tipo y otro:

Análisis de una respuesta de tipo A:

```
else if(type_r == 1){
      memcpy(&aux32b,datagram+sizeRead,sizeof(uint32_t));
      RDATA_q = aux32b;

      memcpy(&antelope.sin_addr, &RDATA_q,sizeof(uint32_t));
      some_addr = inet_ntoa(antelope.sin_addr);
      sizeRead+=RdataLen_r;
```

Análisis de una respuesta de tipo CNAME:

```
if (type_r == 5)
{
      bzero(name_r,TAM);
      bzero(reconstructedName_r, TAM);

n = get_DNS_name(datagram+sizeRead,datagram,reconstructedName_r);
      sizeRead = sizeRead + n;

      var = DNS2domain(reconstructedName_r, name_r);
```

En el primer análisis, se puede apreciar que se establece un filtro si la respuesta es de tipo 1, esto significa que es de tipo A, lo que se traduce como *Adress* (dirección); después a la hora de analizar este tipo de respuestas se recurre a las siguientes librerías de C para traducir las direcciones:

- sys/socket.h
- netinet/in.h
- arpa/inet.h

La función *inet_ntoa()* convierte la dirección de host de Internet, dada en orden de bytes de red, a una cadena en notación decimal punteada IPv4. La cadena se devuelve en un búfer estáticamente asignado, que las llamadas subsiguientes sobrescribirán. Finalmente, en *some_addr* se almacena la cadena que será la que se exporte como resultado.

En cambio, el caso de análisis de las respuestas tipo 5, que significan CNAME difiere del anterior. Este, es un tipo de registro de recurso en el Sistema de Nombres de Dominio (DNS) utilizado para especificar que un nombre de dominio es un alias para otro dominio (el dominio "canónico").
Para ello, se utiliza de nuevo la función *get_DNS_name* que se encarga de reconstruir el nombre y devolvernos la longitud de caracteres que se ha leído por si hubiera más de una respuesta. Para no pisar nombres o evitar fallos por sobreescritura de variables, se realiza un vaciado del búfer con la función *bzero*.
Como se ha podido comprobar en las imágenes anteriores, para obtener finalmente el nombre DNS que se esté analizando en cada momento se utiliza la función: *DNS2Domain*. Esta función convierte un nombre en formato DNS (3www3uam2es0) a un nombre de dominio estándar (www.uam.es). Al igual que la función *get_DNS_name*, esta función devuelve la longitud del nombre DNS para poder seguir llevando la cuenta de los bytes que se llevan leídos en los campos de longitud variable y el nombre de dominio en formato estándar, el cual será finalmente el exportado.

4.5 Conclusión

Tras haber desarrollado las herramientas descritas, se conoce como el hecho de que distintos servidores exporten los datos de distinta forma afecta en gran medida al análisis de datos a bajo nivel. En cuanto al protocolo HTTP, se presenta una herramienta que es capaz de encontrar flujos simétricos que contienen información acerca de la transacción realizada en el momento de estudio y exporta los datos de los flujos que en este TFG se desea que sean objeto de estudio. Estos datos, serán comparados en la fase de validación con los obtenidos mediante la herramienta descrita previamente: *Detect Pro*.
De igual forma se pretende operar con el protocolo DNS, se ha conseguido extraer información valiosa para la obtención de resultados y validación de este TFG. Se consigue, gracias al estudio a nivel de byte, conocer a la perfección el funcionamiento y la organización de la cabecera DNS.
En ambos casos, se pretende que estas herramientas tengan un ciclo de vida útil y que ambas sean actualizables para mejorar su rendimiento y sus prestaciones. A nivel global se ha conseguido llevar a cabo un trabajo de análisis con dos ficheros abiertos en paralelo entrelazados por campos específicos adquiriendo de esta forma, conocimientos sobre programación y estructuración que antes no se conocían. Finalmente, la resolución de grandes tareas mediante segmentación en tareas más pequeñas ha contribuido a una metodología de trabajo constante y ordenada.

5 Integración, pruebas y resultados

5.1 Introducción

A lo largo de este capítulo, se verán los resultados obtenidos por las herramientas desarrolladas, estos se presentan como comparación con los obtenidos por herramientas ya existentes. Con este capítulo, no se pretende evaluar o clasificar una herramienta como mala o buena, sino que se presentarán los resultados para analizar su semejanza con los que ya se han obtenido en otros proyectos ajenos a este trabajo.

Se debe destacar que la realización de esta sección hubiera sido imposible sin las aportaciones de los registros de red realizados por la cátedra UAM Naudit. Es por esto que, a la hora de mostrar los resultados, en lugar de representar las preguntas DNS por su nombre se representan por P1, P2, P3, etc. Para mantener el anonimato de los datos tratados.

5.2 Extracción de resultados

Para realizar la extracción de resultados, se ha seguido la misma estrategia para todas las herramientas. En primer lugar, se utiliza lenguaje AWK el cual es específico para procesar datos basados en texto ya sean ficheros o flujos de datos. Las siglas AWK derivan de los apellidos de los desarrolladores de este lenguaje: Alfred **A**ho, Peter **W**einberger y Brian **K**ernighan.

Mediante una simple sentencia AWK que a continuación se presenta, se iba realizando la extracción de los datos deseados para cada una de las herramientas:

```
$ cat output_dns.txt | awk -F' ' '{ a[$8]++ } END { for (fr in a) {
print fr, a[fr]} }'
```

A continuación, se detalla el comportamiento de la instrucción escrita para que se comprenda mejor el fin que esta tiene:

- Primero, se selecciona el archivo que se quiere leer. En el ejemplo, el archivo bajo estudio es '*output_dns.txt*'.
- Sobre el archivo, se desean seleccionar los datos que estén en la columna número 8. La opción -F' ', indica que se escogerá como separador para el análisis un espacio en blanco.
- Finalmente, mediante la instrucción *for*, se imprime por pantalla el dato escogido y a su lado, el número de veces que ha aparecido.
- Aunque en el ejemplo no aparece, mediante ">>" se exportan los datos impresos por pantalla a un archivo externo. Durante este proyecto, se exportaban los datos obtenidos a archivos con extensión ".csv" para realizar un análisis gráfico de ellos en Excel. Si posteriormente era necesario realizar operaciones en Matlab, mediante la función *xlsread*, se podían importar los datos para realizar operaciones.

Para realizar y verificar los resultados, se han llevado a cabo múltiples comparaciones entre las herramientas que estudian HTTP (HTTP *Flow Alyzer* y HTTP *Dissector*) y las que estudian el protocolo DNS (DNS *Flow Analyzer* y *Parser* DNS).

5.3 Comparación entre resultados

En la siguiente página, se presenta a modo de validación del trabajo realizado una comparación que se ha llevado a cabo sobre los resultados extraídos anteriormente. Existen otros muchos campos de los que se pueden extraer estadísticas relevantes pero la extensión del trabajo permite centrarse en estadísticas particulares. Se seguirá la misma estructura que en la sección anterior para el estudio de los resultados y se debe dejar claro que cuando se habla de porcentaje relativo al total, es sobre el total de transacciones vistas por cada herramienta.

Al igual que se hizo anteriormente, la extracción de datos para mostrar los resultados obtenidos se realiza mediante AWK. Con instrucciones muy similares a la descrita en la sección 5.2 para exportar los datos a archivos con extensión ".csv" y que pudieran ser tratados posteriormente en Excel.

5.3.1 Comparación: HTTP *Flow Analyzer Vs.* HTTP *Dissector*

Para este caso de validación se ha llevado a cabo una limpieza de los resultados obtenidos por el HTTP *Flow Analyzer* para eliminar el método CCM_POST característico de Microsoft. En primer lugar, se muestra la FIGURA 5-1 que detalla los códigos de respuesta vistos por cada una de las dos herramientas en valor absoluto, y en segundo lugar se muestra la FIGURA 5-2 como comparación en porcentaje relativo al total:

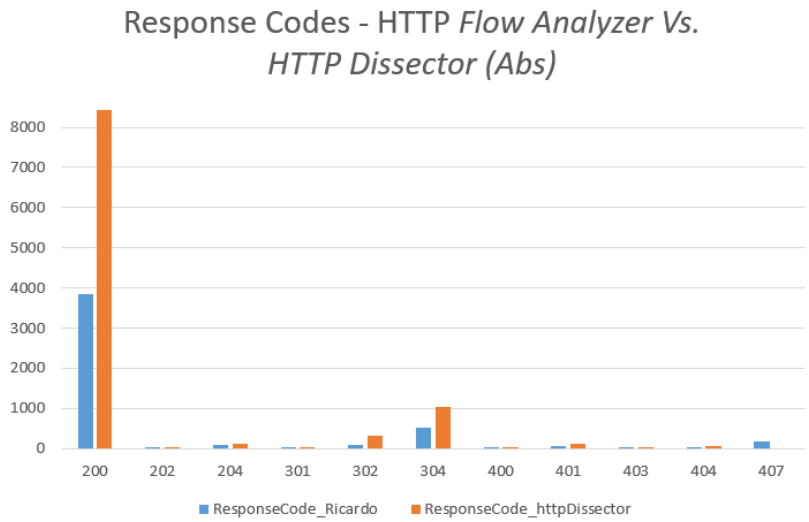

FIGURA 5-1: Comparación de los códigos de respuesta vistos en valor absoluto

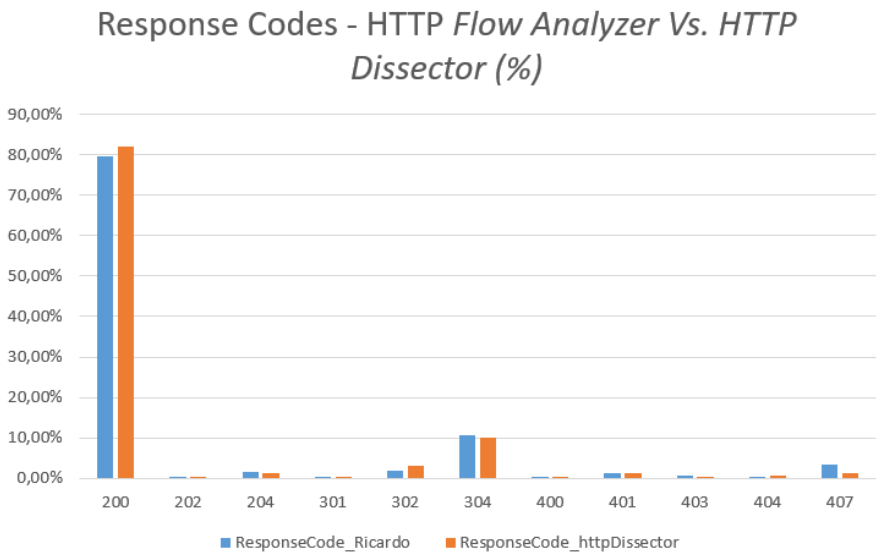

FIGURA 5-2: Comparación entre los códigos de respuesta vistos en porcentaje relativo al total

Se puede notar como el HTTP *Dissector* analiza un número mayor de transacciones HTTP, pero en cuanto a la comparación realizada con el porcentaje relativo al total de transacciones vistas por cada una de las herramientas, la distribución de los códigos de respuesta vistos es similar.

A continuación, se puede ver el número de transacciones realizadas por los puertos de los que se ha obtenido información relativa al protocolo HTTP. En las dos imágenes que se muestran a continuación se puede apreciar, en valor absoluto en la FIGURA 5-4 y en porcentaje relativo al total de transacciones vistas en la FIGURA 5-3, la distribución sobre los puertos:

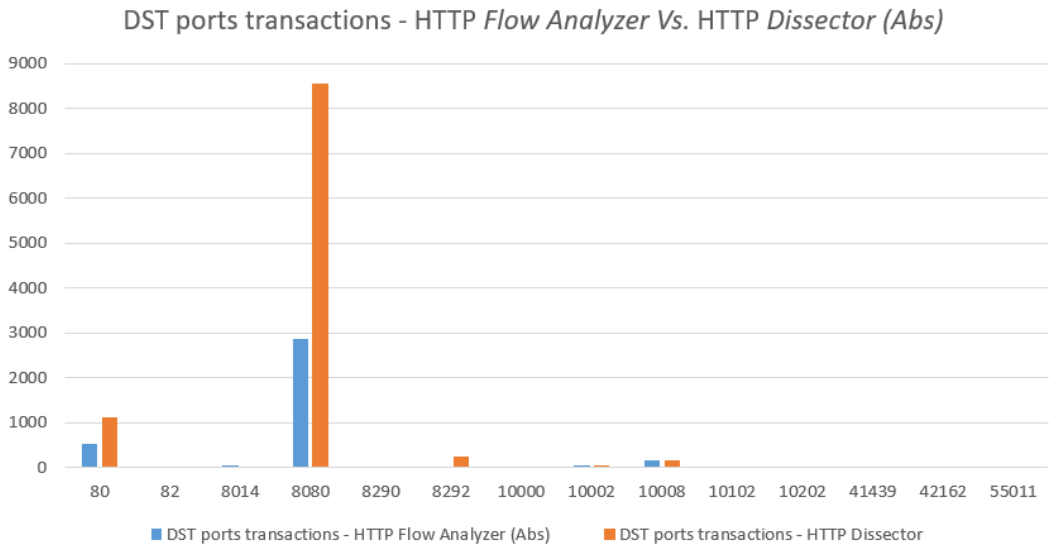

FIGURA 5-4: Comparación de los puertos de destino vistos en valor absoluto

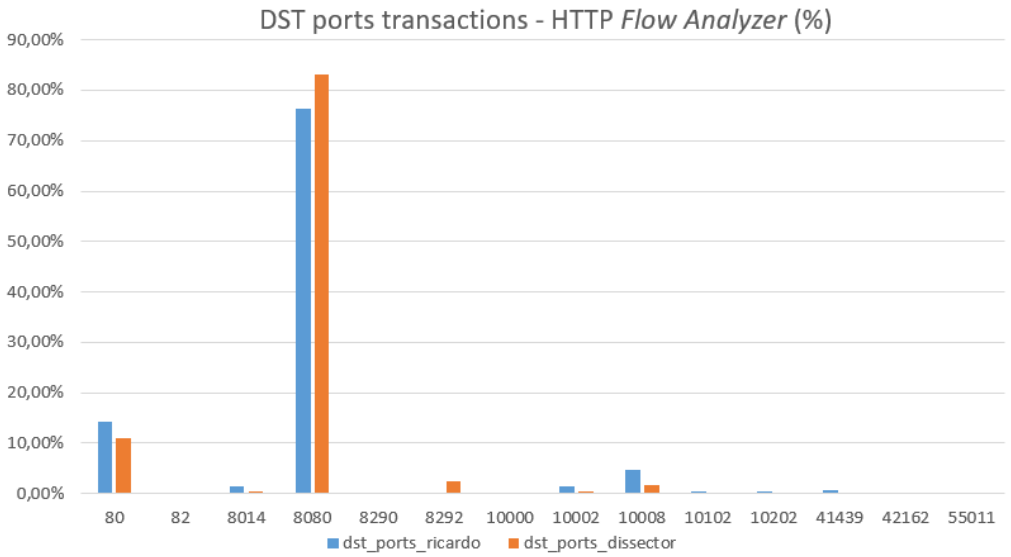

FIGURA 5-3: Comparación de los puertos de destino vistos en porcentaje relativo al total

Finalmente, se presenta una comparación entre los métodos vistos por cada una de las herramientas. Siguiendo la misma metodología de operación que anteriormente se muestran dos figuras: la FIGURA 5-5 contiene el número de métodos vistos en valor absoluto y, la FIGURA 5-6 contiene el porcentaje relativo al total de los métodos vistos:

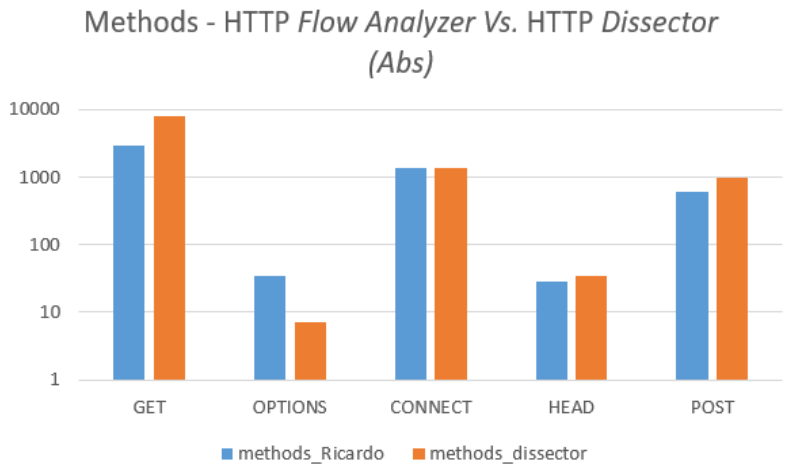

FIGURA 5-5: Comparación de métodos vistos en valor absoluto

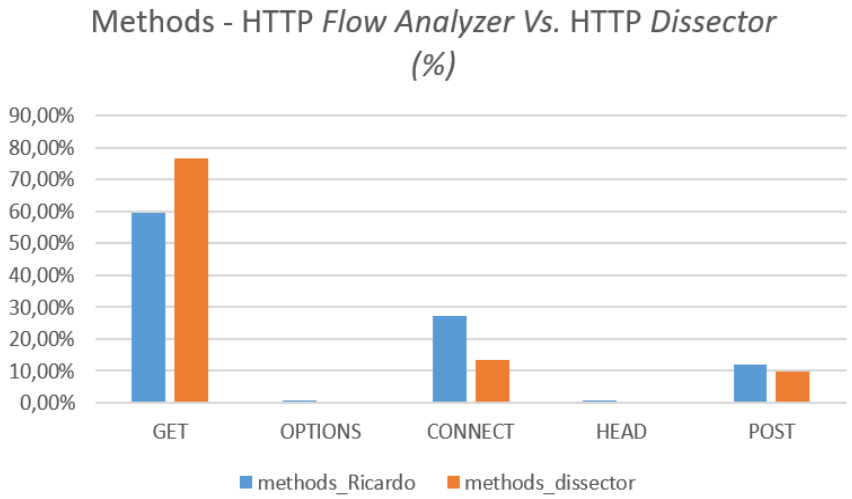

FIGURA 5-6: Comparación de métodos vistos en porcentaje relativo al total

Dada la diferencia entre el número de veces que aparece cada método, se ha decidido establecer una escala logarítmica para visualizar mejor los resultados obtenidos. El hecho de que, en porcentaje relativo al total, la influencia de los CONNECT sea mayor en el HTTP *Flow Analyzer* se debe a que solo se están viendo las peticiones iniciales de cada conexión de manera que si se realiza una petición a través de un proxy, el HTTP *Flow Analyzer* solo será capaz de ver el inicio de la conexión, provocando que el número de métodos GET visto, sea menor.

5.3.2 Comparación: DNS *Flow Analyzer Vs. Parser DNS*

Es en esta sección cuando se realiza una validación sobre la herramienta desarrollada para analizar el protocolo DNS. En esta sección, se comienza realizando una comparación entre los tipos de preguntas DNS vistas por cada una de las herramientas.

En primer lugar, se muestra la FIGURA 5-7 que contiene la información detallada anteriormente en porcentaje relativo al total y seguido de esta, la FIGURA 5-8 en la que se puede ver la misma información, pero en calor absoluto del total de transacciones vistas:

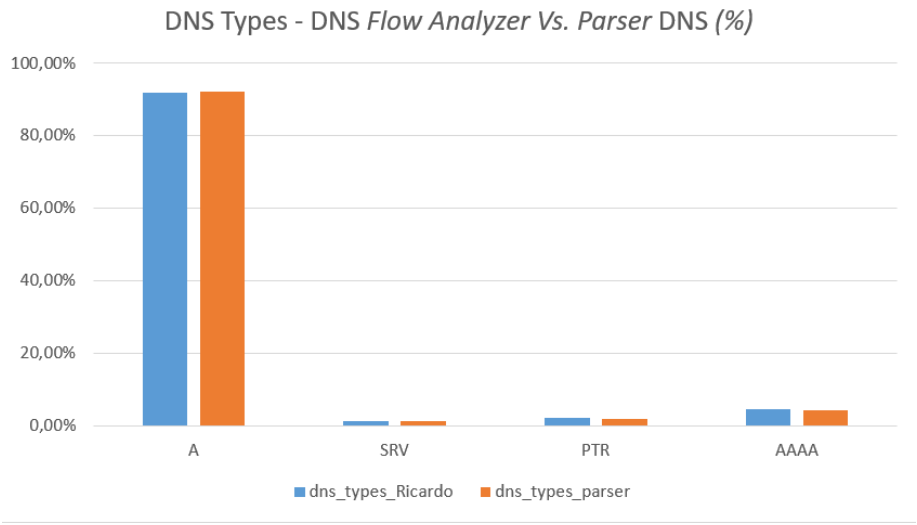

FIGURA 5-7: Comparación de los tipos de preguntas DNS vistas en porcentaje relativo al total

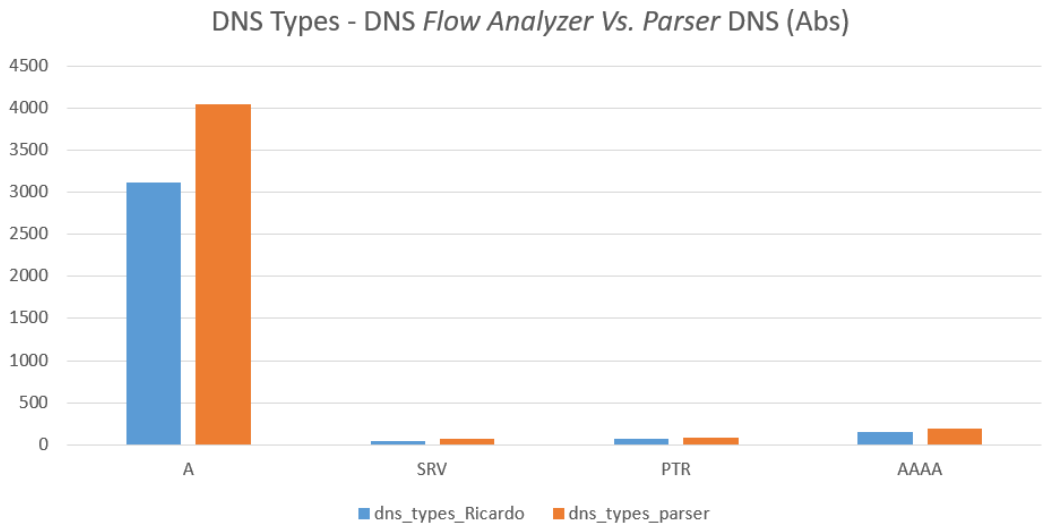

FIGURA 5-8: Comparación de los tipos de preguntas DNS vistas en valor absoluto

En segundo lugar, se representa la comparación sobre cuáles han sido las preguntas que más frecuentemente se han realizado, (de igual forma que en la sección anterior, se dice de una pregunta frecuente cuando esta aparece un porcentaje relativo al total de preguntas superior al 1%). Las preguntas que cumplen el filtro acordado se presentan a continuación tanto en porcentaje relativo al total en la FIGURA 5-10 como en valor absoluto en la FIGURA 5-9:

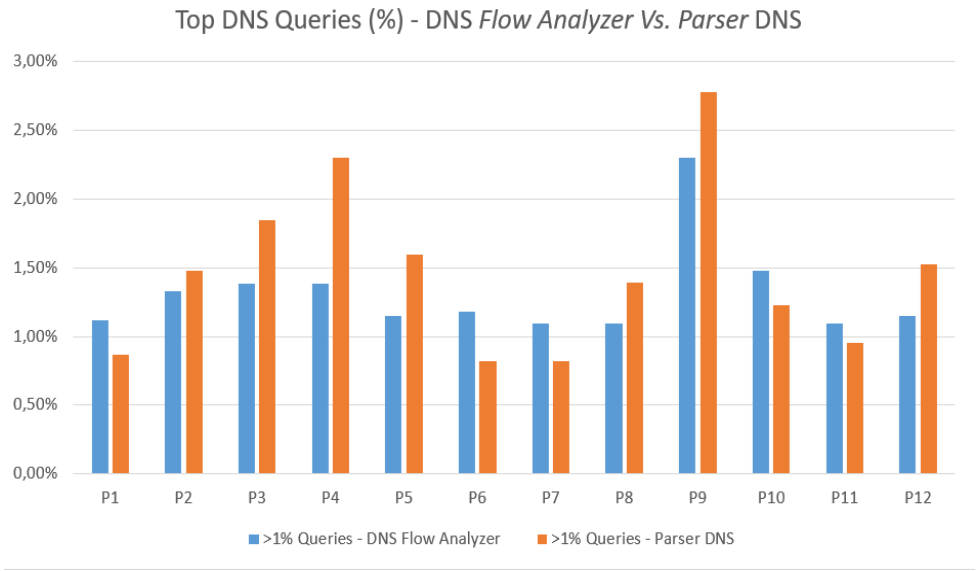

FIGURA 5-10: Comparación entre las preguntas que suponen más del 1% de las transacciones totales en porcentaje relativo al total

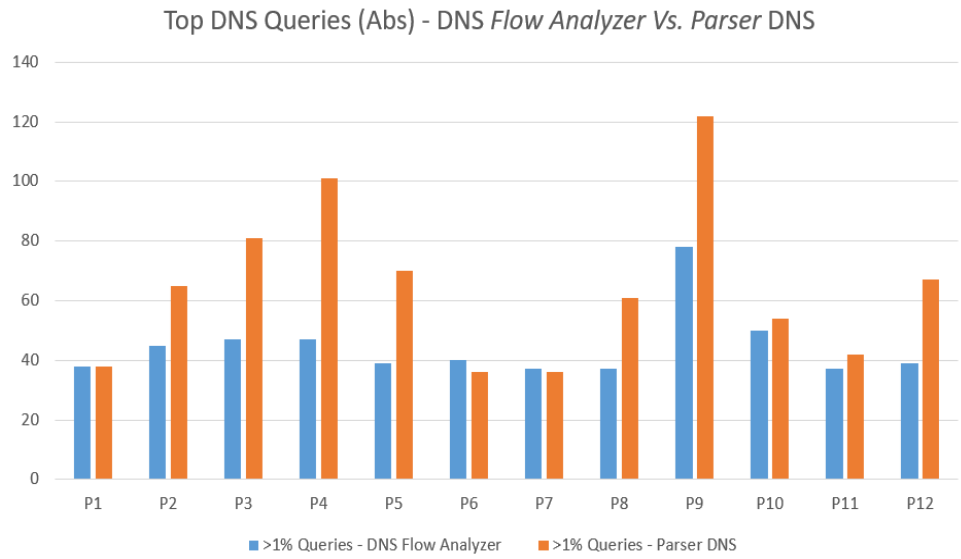

FIGURA 5-9: Comparación entre las preguntas que suponen más del 1% de las transacciones totales en valor absoluto

Además, se puede realizar un análisis complementario sobre las direcciones IP de destino. Con esto, se quiere decir que se muestran a continuación las IP de destino por las que se han realizado más del 1% de las transacciones DNS presentes. Primero, se muestra en la FIGURA 5-11 en valor absoluto y posteriormente se muestra en la FIGURA 5-12 en porcentaje relativo al total de transacciones DNS vistas:

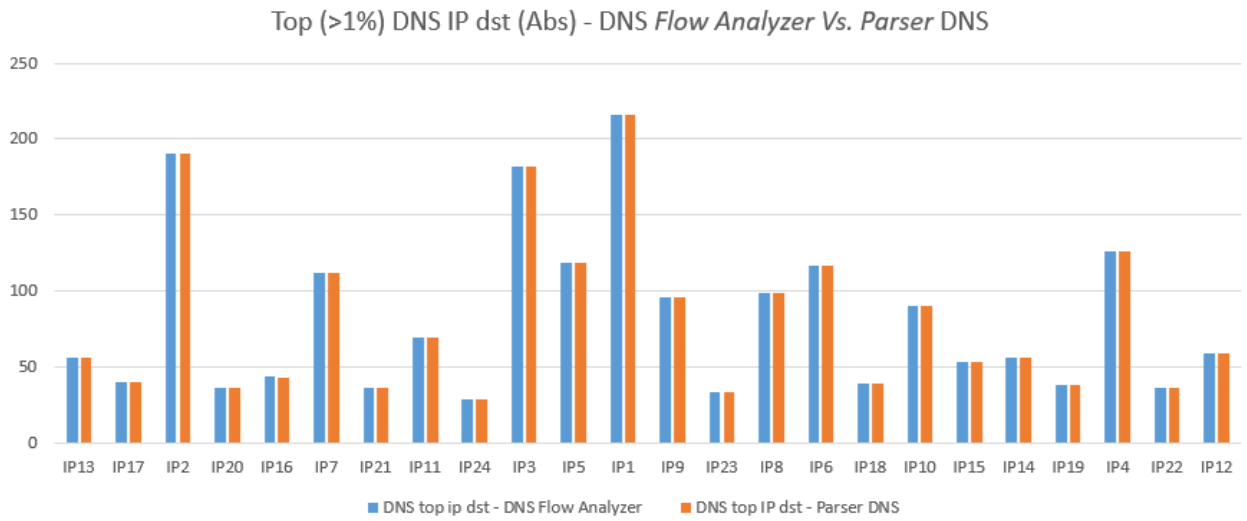

FIGURA 5-11: Comparación entre las direcciones IP de destino que suponen más de un 1% total del tráfico en valor absoluto

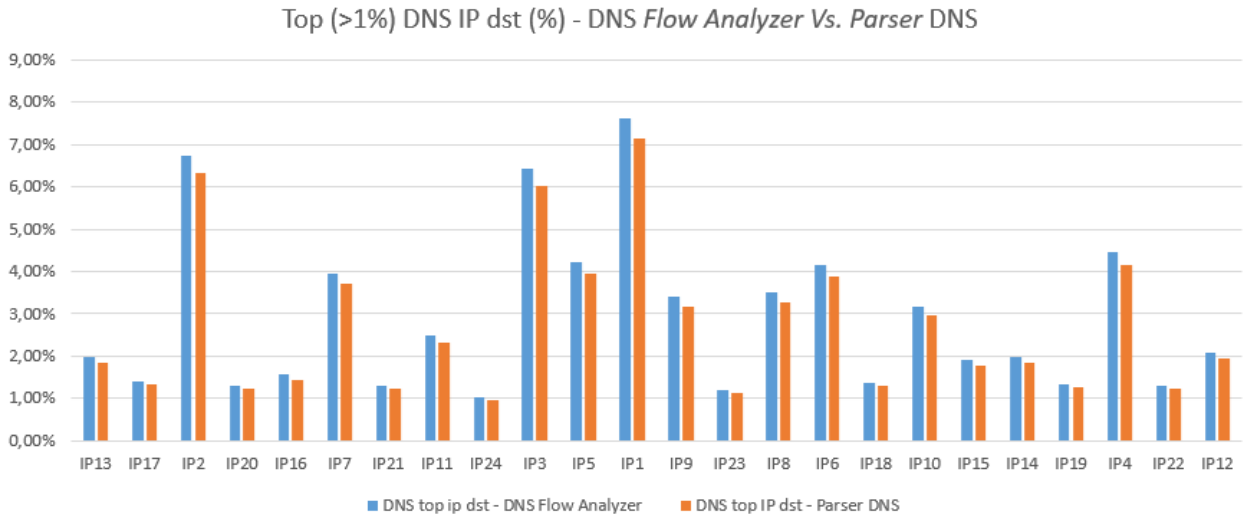

FIGURA 5-12: Comparación entre las direcciones IP de destino que suponen más de un 1% total del tráfico en porcentaje relativo al total

Por último, se presenta en la FIGURA 5-13, la comparación de las CDF (*Cumulative Distribution Function*) de ambas herramientas. Esta representación es relativa a los tiempos de respuesta obtenidos durante la extracción de resultados:

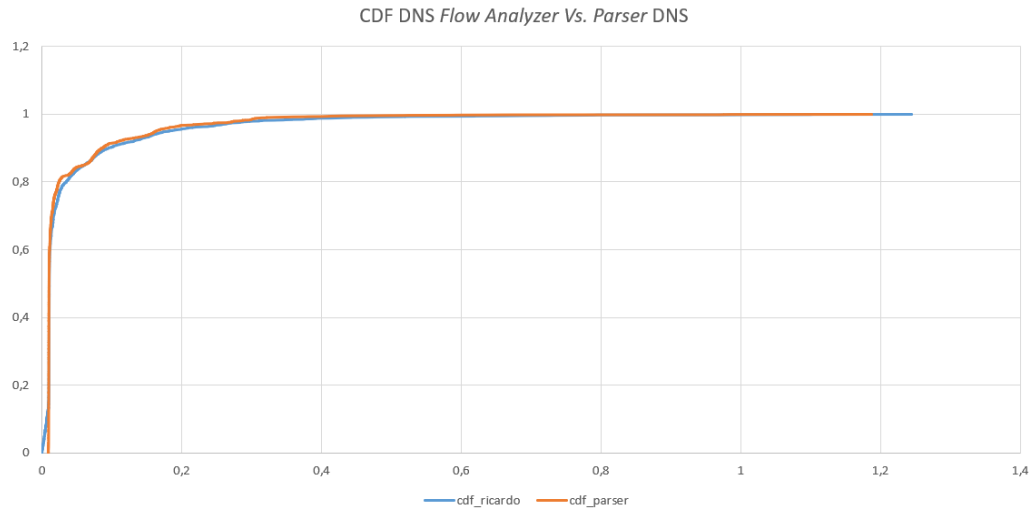

FIGURA 5-13: Comparación de distribuciones CDF respectivas a los tiempos de respuesta de cada herramienta

Esta distribución, representa la similitud existente entre los tiempos de respuesta estudiados en el protocolo DNS. Se puede apreciar como en los extremos las distribuciones son distintas, esto se debe a que los tiempos de respuesta analizados por DNS *Flow Analyzer* son algo distintos en distribución a los captados por *Parser* DNS. Se puede apreciar también una pequeña variación en torno al percentil 8.

5.4 Conclusión

Tras haber visto los resultados extraídos se puede comprobar como el HTTP *Dissector* y el *Parser* DNS son capaces de analizar en valor absoluto un número mayor de flujos. Pero, en cuanto al estudio del porcentaje total sobre las herramientas, se pueden apreciar similitudes que indican que ambas herramientas generan información similar.
Este enfoque hace que no se compare una herramienta frente a otra para ver cuál es mejor, sino que se pretende validar el trabajo realizado durante este proyecto verificando la similitud de la información recopilada en los dos casos de estudio: protocolo HTTP y protocolo DNS.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Durante el tiempo que se ha estado trabajando en este proyecto, no solo se ha aprendido a analizar registros de flujos de red. Además, se ha programado a bajo nivel realizando una ardua tarea de análisis que ha desembocado en la adquisición de nuevos conocimientos sobre herramientas de *parseo* de datos en C. Se ha conseguido enlazar los resultados obtenidos en este lenguaje con otros archivos, mediante sentencias de AWK para poder tratar los resultados en otro tipo de programas.

Se ha aprendido a llevar una metodología de trabajo constante que ha permitido cumplir con los objetivos iniciales gracias a un seguimiento del trabajo por parte del tutor. Se ha conseguido fraccionar un proyecto que parece tener dimensiones demasiado grandes como para que una persona sola por sí misma pueda llevarlo a cabo. Así, se ha visto que estudiando y centrándose en cada momento del proyecto en puntos particulares es posible acabarlo teniendo en cuenta los objetivo marcados. Esta metodología de trabajo, se aprende en la asignatura de Proyectos y Sistemas de Telecomunicación. Gracias a ella, se consigue estructurar de mejor forma la cabeza y, así, poder tener los objetivos claros en cada momento. Se ha podido comprobar como, con los registros de flujos de red, se pueden extraer estadísticas que pueden ser de verdadera utilidad para la monitorización de redes y para optimizar el funcionamiento que estas tienen.

Se han desarrollado los conocimientos adquiridos durante el grado, de forma que se han utilizado estrategias para plantear los análisis que ya se utilizaban cuando se estudiaban asignaturas como Arquitectura de Redes I, Arquitectura de Redes II o Tratamiento Digital de Señales (TSM). Sin las asignaturas que tienen que ver con la rama telemática, el desarrollo de este TFG hubiera sido increíblemente costoso porque cuando se cursaron, se vieron, tanto en prácticas como en teoría, situaciones parecidas a la que en este proyecto se plantean: el hecho de realizar una tarea amplia sobre un campo técnico y ser capaz de dividirla en semanas de trabajo consecutivas para alcanzar el objetivo. En TSM fue la primera vez que se le dedicó un tiempo considerable al análisis de datos. Esta asignatura, aunque sea cursada en otro lenguaje de programación (Matlab), ha sido verdaderamente útil a la hora de realizar cualquier análisis de datos, ya que ayuda a tener muchas variables en consideración al mismo tiempo. Como no puede ser de otra manera, sin haber cursado Programación II que proporciona nociones de programación en C, el desarrollo de este proyecto habría sido mucho más tedioso, debido a que no se conocería a fondo el lenguaje de programación elegido para llevar a cabo este TFG.

6.2 Trabajo futuro

En un futuro, se puede retomar este trabajo reacondicionando el estilo de programación para optimizar el proceso mediante el cual se analizan los flujos. En este aspecto, puede ser de gran utilidad, estudiar en un futuro el rendimiento que las herramientas desarrolladas tienen. Ya que, durante este proyecto el foco se encontraba centrado en el alcance de resultados. También se puede llevar a cabo un análisis de los datos con herramientas que ofrezcan una capacidad de explotación mejor, como *ElasticSearch*, referenciada en [19]. Este tipo de herramientas de terceros, proporcionan una interfaz gráfica capaz de dar forma a los resultados obtenidos en este proyecto. Esto, puede ser verdaderamente útil a medida que

aumente el volumen de datos estudiado, ya que, Excel tiene una limitación en cuanto celdas y, esto puede suponer un conflicto cuando los datos sean exportados.
Además, se pueden estudiar otros protocolos que no ha sido posible abarcar en este proyecto para ahondar en el análisis de flujos tales como ICMP o SSL. De esta forma, se podría sacar partido a información que puede convertirse en provechosa pero que muchas veces pasa desapercibida.

Referencias

[1] The Internet Society, ***Hypertext Transfer Protocol -- HTTP/1.1***, Dirección web: https://tools.ietf.org/pdf/rfc2616.pdf, June 1999

[2] James F. Kurose, Keith W. Ross, ***Redes de computadoras. Un enfoque descendente***, Pearson, 5ª Ed.

[3] P. Mockapetris, Network Working Group, ***Domain Names – Concepts and Facilities***, https://tools.ietf.org/pdf/rfc1034.pdf, November 1987

[4] Richard Stevens, ***TCP/IP Ilustrated***, Dirección Web: http://www.pcvr.nl/tcpip/dns_the.htm, "Chapter 14. DNS: The Domain Name System"

[5] Rick Hofstede, Pavel Celeda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras, ***Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX***, IEEE Communication Surveys & Tutorial, VOL. 16, NO.4, Fourth Quarter 2014, Dirección web: https://ieeexplore.ieee.org/document/6814316/

[6] Flowmon, "***THE MOST POWERFUL NETFLOW/IPFIX EXPORTERS IN THE WORLD***", Dirección Web: https://www.flowmon.com/en/products/flowmon/probe, Último acceso: Junio 2018.

[7] Solarwinds, ***NetFlow Traffic Analyzer***, https://www.solarwinds.com/netflow-traffic-analyzer, Último acceso: Junio 2018

[8] Pere Barlet-Ros, ***On the challenges of network traffic classification with NetFlow/IPFIX***, Dirección Web: https://ripe76.ripe.net/wp-content/uploads/presentations/12-pbarlet-traffic-classification-RIPE76.pdf, Último acceso: Junio 2018

[9] Victor Moreno, Pedro M. Santiago del Río, Javier Ramos, David Muelas, José Luis García-Dorado, Francisco J. Gomez-Arribas and Javier Aracil, ***Multi-granular, multi-purpose and multi-Gb/s monitoring on off-the-shelf systems***, International Journal of Network Management, Dirreción Web: https://onlinelibrary.wiley.com/doi/epdf/10.1002/nem.1861, 5 Junio 2014

[10] Carlos Vega, Paula Roquero, Javier Aracil, ***Multi-Gbps HTTP Traffic Analysis in Commodity Hardware Based on Local Knowledge of TCP Streams***, Computer Networks (2017), doi: 10.1016/j.comnet.2017.01.001, Dirección Web: https://arxiv.org/pdf/1701.04617.pdf , Último acceso: Junio 2018

[11] Kevin R. Fall, W. Richard Stevens, "***TCP/IP Illustrated, Volume 1: The Protocols, Second Edition***", Addison-Wesley Professional, November 2011

[12] David Gourley; Brian Totty; Marjorie Sayer; Anshu Aggarwal; Sailu Reddy, "***HTTP: The Definitive Guide***", O'Reilly Media, Inc., September 2002

[13] IANA (**Internet Assigned Numbers Authority**), https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml, Última actualización: 2017-10-13. Último acceso: Junio 2018.

[14] W. Richard Stevens, ***TCP/IP Ilustrated Volume 1: The Protocols***, Addison-Wesley Professional, December 1993

[15] From Wikipedia, ***the free encyclopedia***, Regular Expression, Dirección Web: https://en.wikipedia.org/wiki/Regular_expression, Última edición: Junio 2018, Último acceso: Junio 2018

[16] Internet World Stats, ***Usage and Population Statistics***, Dirección Web: https://internetworldstats.com/stats.htm, Último acceso: Junio 2018

[17] P. Vixie, Network Working Group, ***Extension Mechanisms for DNS (EDNS0)***, Dirección Web: https://tools.ietf.org/html/rfc2671, Agosto 1999
[18] Talaia, ***Network Intelligence, DDoS and Threat Mitigation***, Dirección Web: https://www.talaia.io/, Último Acceso: Junio 2018
[19] Elastic, ***The Elastic Stack***, Dirección Web: https://www.elastic.co/products, Último acceso: Junio 2018
[20] Solarwinds, ***What are the requirements for NTA 4.x***, Dirección Web: https://support.solarwinds.com/Success_Center/Netflow_Traffic_Analyzer_(NTA)/What_are_the_requirements_for_NTA_4.x, Último acceso: Junio 2018

Anexos

A Estructura detallada sobre los campos de flujos de red.

Número	Campo	Información
1	IP Origen	Dirección IP de origen
2	IP Destino	Dirección IP de destino
3	MAC Origen	Dirección MAC de origen
4	MAC Destino	Dirección MAC de destino
5	Protocolo de transporte	Protocolo de transporte definido por IANA
6	Puerto Origen	Puerto de origen
7	Puerto Destino	Puerto de destino
8	Número de paquetes	Número de paquetes del flujo
9	Número de bytes	Número de bytes del flujo
10	TS Inicio	Timestamp de inicio del flujo
11	TS Fin	Timestamp de fin de flujo
12	Duración	Duración desde inicio hasta final
13	Tamaño máximo de paquete	Tamaño del paquete que más espacio ocupa
14	Tamaño mínimo de paquete	Tamaño del paquete que menos espacio ocupa
15	Tamaño medio de paquete	Tamaño medio de los paquetes
16	Desviación estándar del tamaño de paquete	Desviación estándar de los paquetes
17	Máximo Inter-Arrival	Máximo Inter-Arrival
18	Mínimo Inter-Arrival	Mínimo Inter-Arrival
19	Inter-Arrival Medio	Media de los Inter-Arrival
20	Desviación estándar Inter-Arrival	Desviación estándar de los Inter-Arrival
21	RTT	*Round Trip Time* estimado desde SYN hasta SYN-ACK
22	FIN	Banderas TCP
23	SYN	Banderas TCP
24	RST	Banderas TCP
25	PSH	Banderas TCP
26	ACK	Banderas TCP
27	URG	Banderas TCP
28	CWR	Banderas TCP
29	ECE	Banderas TCP
30	Número de ventanas 0	Ventanas a cero del flujo
31	Número de bytes de payload	Indica el número de bytes a leer del archivo binario
32	Número de bytes de offset	Indica la posición de lectura sobre el arvhivo binario
33	Etiqueta VLAN	Etiqueta *Virtual Local Area Network*
34	Retransmisiones TCP	Retransmisiones TCP